CEP - UP - UTFPR - PUCPr - UFPR –
03/06/2024 - 13:31:33.3
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V:
2.02 76506 CARLOS EDUARDO ANTAO
24CC2204 - 1 _____ / _____ / _____

Executando programas

Um programa é uma unidade de trabalho sendo executado em um computador. Em geral, um programa obtém dados de entrada, processa-os e gera dados de saída. Um detalhe importante é que (graças à arquitetura de Von Neumann) programas são tratados e armazenados na memória como se fossem dados.

A CPU utiliza ciclos de máquina repetitivos para executar as instruções do programa, uma a uma, do início ao fim. Cada ciclo é composto pelas fases de busca, decodificação e execução. Na fase de busca, a unidade de controle manda o sistema copiar no registrador de instruções a próxima instrução. (isto é feito a partir do registrador chamado contador do programa). Ao acabar esta fase o contador é incrementado para apontar para a próxima instrução. A fase de decodificação visa preparar os circuitos necessários para obedecer à instrução, seja ela qual for. Finalmente, a execução, é a implementação da instrução.

Note que as velocidades da CPU e dos dispositivos de E/S são completamente díspares, o que implica em algum tipo de sincronização (e obviamente abre as portas para inúmeros níveis de multiprocessamento). Três métodos são usados para obter sincronização: E/S programada (o mais primitivo: a CPU espera pelo dispositivo de E/S), E/S dirigido por interrupção, onde a CPU comanda a transferência, mas não fica aguardando sua conclusão. Aqui o dispositivo de E/S interrompe a CPU quando acabar. Durante este tempo, a CPU pode realizar outras tarefas. Finalmente, o método de Acesso Direto à memória (DMA), transfere um grande bloco de dados entre o dispositivo (rápido) de E/S e a memória, sem precisar passar pela CPU. Isto requer um controlador de DMA.

CPUS modernas
A seguir algumas descrições de processadores modernos, no estado da arte em 2024:

- CPU Intel Core i9-12900K: 5,5 GHz, 16 núcleos, 24 threads, até 5,5 GHz de clock boost, taxa de ciclo de máquina estimada entre 220 e 275 ciclos por instrução (CPI).
- CPU AMD Ryzen 9 5950X: 4,9 GHz, 16 núcleos, 32 threads, até 5,0 GHz de clock boost, taxa de ciclo de máquina estimada entre 200 e 250 CPI.
- CPU Apple M1 Pro: 3,2 GHz, 10 núcleos (8 núcleos de alto desempenho + 2 núcleos de alta eficiência), 16 threads, até 3,2 GHz de clock boost, taxa de ciclo de máquina estimada entre 150 e 200 CPI.

A taxa de ciclo é uma medida da eficiência ao executar uma instrução: ela indica o número de ciclos de relógio necessários para executar 1 instrução. O clock boost é uma execução acelerada que pode ser invocada pela CPU (depende de temperatura, carga, eficiência energética, etc)

Um núcleo é capaz de executar uma instrução por vez. Cada núcleo possui seus próprios recursos dedicados, como registradores e unidades de execução, permitindo que ele processe tarefas de forma independente. Quanto mais núcleos uma CPU possui ela pode lidar com mais tarefas simultaneamente.

As threads, por outro lado, dividem o tempo de processamento com os núcleos. Elas compartilham os mesmos recursos do núcleo, mas podem alternar a execução de diferentes instruções. Isso permite que um único núcleo execute várias threads ao mesmo tempo, aumentando ainda mais a capacidade multitarefa da CPU.

Uma analogia: Imagine uma cozinha como a CPU e os chefs como os núcleos. Cada chef (núcleo) tem sua própria estação de trabalho (recursos dedicados) e pode preparar um prato (instrução) por vez. As garçonetes (threads) circulam pela cozinha, pegando os pratos prontos dos chefs e entregando-os aos clientes (aplicativos). Quanto mais chefs (núcleos) e garçonetes (threads) a cozinha tiver, mais pratos (tarefas) ela poderá preparar e servir (executar) ao mesmo tempo.

Diferentes arquiteturas

CISC acrônimo de *Complex Instruction Set Computer*. Tem um grande conjunto de instruções, incluindo muitas complexas. Torna o processo de programação mais fácil, já que para muitas tarefas, há uma instrução nativa. A complexidade das instruções torna os circuitos da CPU e da UC mais complicados. A maneira de lidar com isso foi construir a programação em dois níveis. No primeiro, chamado micro-operações, estão apenas as operações muito simples, as únicas efetuadas diretamente pela CPU. Surge uma micromemória, que contém a tradução das micro-operações da operação complexa que está sendo executada. Como tudo na vida, há vantagens (programas menores no nível da máquina) e desvantagens (sobrecarga da micromemória e da micro-programação). Um exemplo desta arquitetura é a série Pentium da Intel, bem como mainframes e supercomputadores. Note-se que os PCs parecem estar migrando para a arquitetura RISC.

RISC Acrônimo de *Reduced Instruction Set Computer*. A idéia é ter o mínimo possível no conjunto de instruções de máquina, traduzindo operações complexas em macros usando operações simples sob controle do programa (e não da máquina como no caso CISC). Exemplos de computadores RISC: smartphones e tablets (processadores ARM), consoles de videogame (Nintendo Switch), Apple MacBook Air, entre outros.
Para encerrar vale dizer que mais recentemente computadores começaram a usar recursos de ambas as abordagens.

Pipeline Como se viu, um computador executa operações em 3 fases: busca, decodificação e execução. Originalmente, as 3 fases ocorriam em série para cada instrução: a instrução x precisava concluir as 3 fases antes de começar o ciclo de instrução $x+1$. Mais modernamente os computadores começaram a usar a técnica de *pipeline*, permitindo que a UC possa começar as fases da $x+1$ enquanto ainda trata a x.

Processamento paralelo Quando o custo do hardware começou a cair, computadores começaram a ser desenhados com várias UCs, várias ULAs e várias memórias. Isto permite o paralelismo na execução de instruções. Agora surgem 4 possibilidades: SISD (fluxo único de instruções e único de dados), SIMD (fluxo único de instruções e múltiplo de dados), MISD (único de dados e múltiplo de instruções) e MIMD (múltiplos ambos). Vale notar que o MISD vale como conceito, mas parece não ter sido nunca implementado. A idéia do processamento paralelo, parece mais fácil como conceito do que na prática. Há aqui um problema de balanceamento de carga: a alocação dinâmica de tarefas para que estas sejam executadas nos diversos processardores disponíveis, mas de forma que todos os processadores sejam utilizados adequadamente. Associado a isto está o problema do escalonamento, em inglês *scaling* (divisão de uma tarefa em sub-tarefas de acordo com o número de processadores disponíveis).

Com o aumento do número de tarefas, cresce exponencialmente o trabalho necessário para coordenar a interação entre elas. Se houver 4 tarefas, há potencialmente 6 pares de tarefas se comunicando. Em 5 tarefas, há 10 pares e em 6 tarefas, 15 pares.

Filas dentro do Sist. Operacional

Imagine-se um sistema operacional multiprogramado despachando tarefas. Como o processador é um só, e existem muitos processos querendo-o usar, formam-se filas. Alem do algoritmo FIFO também conhecido como FCFS (*fist come, first served*), vão ser usados 2 algoritmos distintos:

SJF *Shortest Job First.* Neste esquema, quando o processador vai atender alguém, ele escolhe na fila de espera, aquele que tiver a menor requisição de tempo. Se houver empate, o primeiro será retirado.

RR *Round Robin.* Neste esquema preemptivo, a fila será atendida em ordem, mas com uma quantidade fixa de tempo para cada processo. Após receber este *quantum*, o processo vai para o fim da fila.

Para este problema, haverá um padrão de chegadas de requisições de tempo. A unidade a ser considerada é o segundo. Não haverá nenhum tempo de *overhead* para troca de processos. Irão ser usados para cada processo, dois tempos, assim definidos:

duração do processo tempo de fim - tempo de início + 1

elapsed time tempo de fim - tempo de chegada + 1

Acompanhe no exemplo feito, usando o método **Shortest Job First**

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
P	3	10	3	12	10	10
L	4	9	58	66	9	63
K	7	7	15	21	7	15
R	10	2	13	14	2	5
T	12	9	67	75	9	64
Y	20	5	24	28	5	9
N	21	2	22	23	2	3
I	25	9	76	84	9	60
C	26	11	85	95	11	70
V	27	8	35	42	8	16
F	29	6	29	34	6	6
S	34	11	96	106	11	73
M	36	8	50	57	8	22
G	40	2	43	44	2	5
Z	44	5	45	49	5	6

Veja o resultado do mapa do processador, onde cada letra corresponde a 1 seg. Espaços são só para clareza e separam 10 segundos.

```
 PPPPPPPP PPRRKKKKKK KNNYYYYYFF FFFFVVVVVV VVGGZ
ZZZZM MMMMMMLLL LLLLLLTTTT TTTTTIIIII IIIICCCCCC
 CCCCCSSSSS SSSSSS
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 6.9 segundos e 28.5 segundos respectivamente.

☞ Para você fazer

Imagine a seguinte situação em um ambiente multiprogramado com o método **Shortest Job First** .

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
T	1	10				
K	4	5				
F	6	4				
V	7	5				
J	11	6				
E	17	2				
A	27	7				
D	29	6				
Y	30	7				
R	31	6				
W	32	10				
G	33	4				
B	37	7				
M	40	8				
L	42	11				

Para preencher a tabela acima, use o seguinte espaço de tempo

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
100	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150

Responda aqui, qual a média de duração e de elapsed do exercício proposto com o método **Shortest Job First** . (1 casa decimal, por favor)

duração	elapsed

204-76506 - /

CEP - UP - UTFPR - PUCPr - UFPR –
03/06/2024 - 13:31:33.3
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V:
2.02 76663 DANILO DEPETRIS SOARES
24CC2204 - 2 _____ / _____ / _____

Executando programas

Um programa é uma unidade de trabalho sendo executado em um computador. Em geral, um programa obtém dados de entrada, processa-os e gera dados de saída. Um detalhe importante é que (graças à arquitetura de Von Neumann) programas são tratados e armazenados na memória como se fossem dados.

A CPU utiliza ciclos de máquina repetitivos para executar as instruções do programa, uma a uma, do início ao fim. Cada ciclo é composto pelas fases de busca, decodificação e execução. Na fase de busca, a unidade de controle manda o sistema copiar no registrador de instruções a próxima instrução. (isto é feito a partir do registrador chamado contador do programa). Ao acabar esta fase o contador é incrementado para apontar para a próxima instrução. A fase de decodificação visa preparar os circuitos necessários para obedecer à instrução, seja ela qual for. Finalmente, a execução, é a implementação da instrução.

Note que as velocidades da CPU e dos dispositivos de E/S são completamente díspares, o que implica em algum tipo de sincronização (e obviamente abre as portas para inúmeros níveis de multiprocessamento). Três métodos são usados para obter sincronização: E/S programada (o mais primitivo: a CPU espera pelo dispositivo de E/S), E/S dirigido por interrupção, onde a CPU comanda a transferência, mas não fica aguardando sua conclusão. Aqui o dispositivo de E/S interrompe a CPU quando acabar. Durante este tempo, a CPU pode realizar outras tarefas. Finalmente, o método de Acesso Direto à memória (DMA), transfere um grande bloco de dados entre o dispositivo (rápido) de E/S e a memória, sem precisar passar pela CPU. Isto requer um controlador de DMA.

CPUS modernas
A seguir algumas descrições de processadores modernos, no estado da arte em 2024:

- CPU Intel Core i9-12900K: 5,5 GHz, 16 núcleos, 24 threads, até 5,5 GHz de clock boost, taxa de ciclo de máquina estimada entre 220 e 275 ciclos por instrução (CPI).
- CPU AMD Ryzen 9 5950X: 4,9 GHz, 16 núcleos, 32 threads, até 5,0 GHz de clock boost, taxa de ciclo de máquina estimada entre 200 e 250 CPI.
- CPU Apple M1 Pro: 3,2 GHz, 10 núcleos (8 núcleos de alto desempenho + 2 núcleos de alta eficiência), 16 threads, até 3,2 GHz de clock boost, taxa de ciclo de máquina estimada entre 150 e 200 CPI.

A taxa de ciclo é uma medida da eficiência ao executar uma instrução: ela indica o número de ciclos de relógio necessários para executar 1 instrução. O clock boost é uma execução acelerada que pode ser invocada pela CPU (depende de temperatura, carga, eficiência energética, etc)

Um núcleo é capaz de executar uma instrução por vez. Cada núcleo possui seus próprios recursos dedicados, como registradores e unidades de execução, permitindo que ele processe tarefas de forma independente. Quanto mais núcleos uma CPU possui ela pode lidar com mais tarefas simultaneamente.

As threads, por outro lado, dividem o tempo de processamento com os núcleos. Elas compartilham os mesmos recursos do núcleo, mas podem alternar a execução de diferentes instruções. Isso permite que um único núcleo execute várias threads ao mesmo tempo, aumentando ainda mais a capacidade multitarefa da CPU.

Uma analogia: Imagine uma cozinha como a CPU e os chefs como os núcleos. Cada chef (núcleo) tem sua própria estação de trabalho (recursos dedicados) e pode preparar um prato (instrução) por vez. As garçonetes (threads) circulam pela cozinha, pegando os pratos prontos dos chefs e entregando-os aos clientes (aplicativos). Quanto mais chefs (núcleos) e garçonetes (threads) a cozinha tiver, mais pratos (tarefas) ela poderá preparar e servir (executar) ao mesmo tempo.

Diferentes arquiteturas

CISC acrônimo de *Complex Instruction Set Computer*. Tem um grande conjunto de instruções, incluindo muitas complexas. Torna o processo de programação mais fácil, já que para muitas tarefas, há uma instrução nativa. A complexidade das instruções torna os circuitos da CPU e da UC mais complicados. A maneira de lidar com isso foi construir a programação em dois níveis. No primeiro, chamado micro-operações, estão apenas as operações muito simples, as únicas efetuadas diretamente pela CPU. Surge uma micromemória, que contém a tradução das micro-operações da operação complexa que está sendo executada. Como tudo na vida, há vantagens (programas menores no nível da máquina) e desvantagens (sobrecarga da micromemória e da micro-programação). Um exemplo desta arquitetura é a série Pentium da Intel, bem como mainframes e supercomputadores. Note-se que os PCs parecem estar migrando para a arquitetura RISC.

RISC Acrônimo de *Reduced Instruction Set Computer*. A idéia é ter o mínimo possível no conjunto de instruções de máquina, traduzindo operações complexas em macros usando operações simples sob controle do programa (e não da máquina como no caso CISC). Exemplos de computadores RISC: smartphones e tablets (processadores ARM), consoles de videogame (Nintendo Switch), Apple MacBook Air, entre outros.
Para encerrar vale dizer que mais recentemente computadores começaram a usar recursos de ambas as abordagens.

Pipeline Como se viu, um computador executa operações em 3 fases: busca, decodificação e execução. Originalmente, as 3 fases ocorriam em série para cada instrução: a instrução x precisava concluir as 3 fases antes de começar o ciclo de instrução $x+1$. Mais modernamente os computadores começaram a usar a técnica de *pipeline*, permitindo que a UC possa começar as fases da $x+1$ enquanto ainda trata a x.

Processamento paralelo Quando o custo do hardware começou a cair, computadores começaram a ser desenhados com várias UCs, várias ULAs e várias memórias. Isto permite o paralelismo na execução de instruções. Agora surgem 4 possibilidades: SISD (fluxo único de instruções e único de dados), SIMD (fluxo único de instruções e múltiplo de dados), MISD (único de dados e múltiplo de instruções) e MIMD (múltiplos ambos). Vale notar que o MISD vale como conceito, mas parece não ter sido nunca implementado. A idéia do processamento paralelo, parece mais fácil como conceito do que na prática. Há aqui um problema de balanceamento de carga: a alocação dinâmica de tarefas para que estas sejam executadas nos diversos processardores disponíveis, mas de forma que todos os processadores sejam utilizados adequadamente. Associado a isto está o problema do escalonamento, em inglês *scaling* (divisão de uma tarefa em sub-tarefas de acordo com o número de processadores disponíveis).

Com o aumento do número de tarefas, cresce exponencialmente o trabalho necessário para coordenar a interação entre elas. Se houver 4 tarefas, há potencialmente 6 pares de tarefas se comunicando. Em 5 tarefas, há 10 pares e em 6 tarefas, 15 pares.

Filas dentro do Sist. Operacional

Imagine-se um sistema operacional multiprogramado despachando tarefas. Como o processador é um só, e existem muitos processos querendo-o usar, formam-se filas. Alem do algoritmo FIFO também conhecido como FCFS (*fist come, first served*), vão ser usados 2 algoritmos distintos:

SJF *Shortest Job First.* Neste esquema, quando o processador vai atender alguém, ele escolhe na fila de espera, aquele que tiver a menor requisição de tempo. Se houver empate, o primeiro será retirado.

RR *Round Robin.* Neste esquema preemptivo, a fila será atendida em ordem, mas com uma quantidade fixa de tempo para cada processo. Após receber este *quantum*, o processo vai para o fim da fila.

Para este problema, haverá um padrão de chegadas de requisições de tempo. A unidade a ser considerada é o segundo. Não haverá nenhum tempo de *overhead* para troca de processos. Irão ser usados para cada processo, dois tempos, assim definidos:

duração do processo tempo de fim - tempo de início + 1

elapsed time tempo de fim - tempo de chegada + 1

Acompanhe no exemplo feito, usando o método **Shortest Job First**

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
S	3	6	3	8	6	6
O	8	2	9	10	2	3
K	12	7	12	18	7	7
N	14	10	23	32	10	19
W	17	10	66	75	10	59
P	18	4	19	22	4	5
X	27	3	33	35	3	9
G	28	10	76	85	10	58
V	30	10	86	95	10	66
Y	31	11	96	106	11	76
J	35	9	48	56	9	22
L	36	4	36	39	4	4
C	38	9	57	65	9	28
Q	39	6	40	45	6	7
A	43	2	46	47	2	5

Veja o resultado do mapa do processador, onde cada letra corresponde a 1 seg. Espaços são só para clareza e separam 10 segundos.

```
  SSSSSSOO  KKKKKKKPP PPNNNNNNNN NNXXXLLLLQ QQQQQ
AAJJJ JJJJJJCCCC CCCCCWWWWW WWWWWGGGGG GGGGGVVVVV
 VVVVVYYYYY YYYYYY
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 6.9 segundos e 24.9 segundos respectivamente.

☞ Para você fazer

Imagine a seguinte situação em um ambiente multiprogramado com o método **Shortest Job First** .

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
B	4	9				
V	5	10				
H	7	9				
S	8	3				
L	13	9				
M	15	11				
Y	16	4				
O	20	9				
E	22	7				
A	28	6				
W	29	2				
I	33	11				
D	34	10				
X	39	3				
K	43	10				

Para preencher a tabela acima, use o seguinte espaço de tempo

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
100	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150

Responda aqui, qual a média de duração e de elapsed do exercício proposto com o método **Shortest Job First** . (1 casa decimal, por favor)

duração	elapsed

204-76663 - /

Executando programas

Um programa é uma unidade de trabalho sendo executado em um computador. Em geral, um programa obtém dados de entrada, processa-os e gera dados de saída. Um detalhe importante é que (graças à arquitetura de Von Neumann) programas são tratados e armazenados na memória como se fossem dados.

A CPU utiliza ciclos de máquina repetitivos para executar as instruções do programa, uma a uma, do início ao fim. Cada ciclo é composto pelas fases de busca, decodificação e execução. Na fase de busca, a unidade de controle manda o sistema copiar no registrador de instruções a próxima instrução. (isto é feito a partir do registrador chamado contador do programa). Ao acabar esta fase o contador é incrementado para apontar para a próxima instrução. A fase de decodificação visa preparar os circuitos necessários para obedecer à instrução, seja ela qual for. Finalmente, a execução, é a implementação da instrução.

Note que as velocidades da CPU e dos dispositivos de E/S são completamente díspares, o que implica em algum tipo de sincronização (e obviamente abre as portas para inúmeros níveis de multiprocessamento). Três métodos são usados para obter sincronização: E/S programada (o mais primitivo: a CPU espera pelo dispositivo de E/S), E/S dirigido por interrupção, onde a CPU comanda a transferência, mas não fica aguardando sua conclusão. Aqui o dispositivo de E/S interrompe a CPU quando acabar. Durante este tempo, a CPU pode realizar outras tarefas. Finalmente, o método de Acesso Direto à memória (DMA), transfere um grande bloco de dados entre o dispositivo (rápido) de E/S e a memória, sem precisar passar pela CPU. Isto requer um controlador de DMA.

CPUS modernas

A seguir algumas descrições de processadores modernos, no estado da arte em 2024:

- CPU Intel Core i9-12900K: 5,5 GHz, 16 núcleos, 24 threads, até 5,5 GHz de clock boost, taxa de ciclo de máquina estimada entre 220 e 275 ciclos por instrução (CPI).
- CPU AMD Ryzen 9 5950X: 4,9 GHz, 16 núcleos, 32 threads, até 5,0 GHz de clock boost, taxa de ciclo de máquina estimada entre 200 e 250 CPI.
- CPU Apple M1 Pro: 3,2 GHz, 10 núcleos (8 núcleos de alto desempenho + 2 núcleos de alta eficiência), 16 threads, até 3,2 GHz de clock boost, taxa de ciclo de máquina estimada entre 150 e 200 CPI.

A taxa de ciclo é uma medida da eficiência ao executar uma instrução: ela indica o número de ciclos de relógio necessários para executar 1 instrução. O clock boost é uma execução acelerada que pode ser invocada pela CPU (depende de temperatura, carga, eficiência energética, etc)

Um núcleo é capaz de executar uma instrução por vez. Cada núcleo possui seus próprios recursos dedicados, como registradores e unidades de execução, permitindo que ele processe tarefas de forma independente. Quanto mais núcleos uma CPU possui ela pode lidar com mais tarefas simultaneamente.

As threads, por outro lado, dividem o tempo de processamento com os núcleos. Elas compartilham os mesmos recursos do núcleo, mas podem alternar a execução de diferentes instruções. Isso permite que um único núcleo execute várias threads ao mesmo tempo, aumentando ainda mais a capacidade multitarefa da CPU.

Uma analogia: Imagine uma cozinha como a CPU e os chefs como os núcleos. Cada chef (núcleo) tem sua própria estação de trabalho (recursos dedicados) e pode preparar um prato (instrução) por vez. As garçonetes (threads) circulam pela cozinha, pegando os pratos prontos dos chefs e entregando-os aos clientes (aplicativos). Quanto mais chefs (núcleos) e garçonetes (threads) a cozinha tiver, mais pratos (tarefas) ela poderá preparar e servir (executar) ao mesmo tempo.

Diferentes arquiteturas

CISC acrônimo de *Complex Instruction Set Computer*. Tem um grande conjunto de instruções, incluindo muitas complexas. Torna o processo de programação mais fácil, já que para muitas tarefas, há uma instrução nativa. A complexidade das instruções torna os circuitos da CPU e da UC mais complicados. A maneira de lidar com isso foi construir a programação em dois níveis. No primeiro, chamado micro-operações, estão apenas as operações muito simples, as únicas efetuadas diretamente pela CPU. Surge uma micromemória, que contém a tradução das micro-operações da operação complexa que está sendo executada. Como tudo na vida, há vantagens (programas menores no nível da máquina) e desvantagens (sobrecarga da micromemória e da micro-programação). Um exemplo desta arquitetura é a série Pentium da Intel, bem como mainframes e supercomputadores. Note-se que os PCs parecem estar migrando para a arquitetura RISC.

RISC Acrônimo de *Reduced Instruction Set Computer*. A idéia é ter o mínimo possível no conjunto de instruções de máquina, traduzindo operações complexas em macros usando operações simples sob controle do programa (e não da máquina como no caso CISC). Exemplos de computadores RISC: smartphones e tablets (processadores ARM), consoles de videogame (Nintendo Switch), Apple MacBook Air, entre outros.
Para encerrar vale dizer que mais recentemente computadores começaram a usar recursos de ambas as abordagens.

Pipeline Como se viu, um computador executa operações em 3 fases: busca, decodificação e execução. Originalmente, as 3 fases ocorriam em série para cada instrução: a instrução x precisava concluir as 3 fases antes de começar o ciclo de instrução $x+1$. Mais modernamente os computadores começaram a usar a técnica de *pipeline*, permitindo que a UC possa começar as fases da $x+1$ enquanto ainda trata a x.

Processamento paralelo Quando o custo do hardware começou a cair, computadores começaram a ser desenhados com várias UCs, várias ULAs e várias memórias. Isto permite o paralelismo na execução de instruções. Agora surgem 4 possibilidades: SISD (fluxo único de instruções e único de dados), SIMD (fluxo único de instruções e múltiplo de dados), MISD (único de dados e múltiplo de instruções) e MIMD (múltiplos ambos). Vale notar que o MISD vale como conceito, mas parece não ter sido nunca implementado. A idéia do processamento paralelo, parece mais fácil como conceito do que na prática. Há aqui um problema de balanceamento de carga: a alocação dinâmica de tarefas para que estas sejam executadas nos diversos processardores disponíveis, mas de forma que todos os processadores sejam utilizados adequadamente. Associado a isto está o problema do escalonamento, em inglês *scaling* (divisão de uma tarefa em sub-tarefas de acordo com o número de processadores disponíveis).

Com o aumento do número de tarefas, cresce exponencialmente o trabalho necessário para coordenar a interação entre elas. Se houver 4 tarefas, há potencialmente 6 pares de tarefas se comunicando. Em 5 tarefas, há 10 pares e em 6 tarefas, 15 pares.

Filas dentro do Sist. Operacional

Imagine-se um sistema operacional multiprogramado despachando tarefas. Como o processador é um só, e existem muitos processos querendo-o usar, formam-se filas. Alem do algoritmo FIFO também conhecido como FCFS (*fist come, first served*), vão ser usados 2 algoritmos distintos:

SJF *Shortest Job First.* Neste esquema, quando o processador vai atender alguém, ele escolhe na fila de espera, aquele que tiver a menor requisição de tempo. Se houver empate, o primeiro será retirado.

RR *Round Robin.* Neste esquema preemptivo, a fila será atendida em ordem, mas com uma quantidade fixa de tempo para cada processo. Após receber este *quantum*, o processo vai para o fim da fila.

Para este problema, haverá um padrão de chegadas de requisições de tempo. A unidade a ser considerada é o segundo. Não haverá nenhum tempo de *overhead* para troca de processos. Irão ser usados para cada processo, dois tempos, assim definidos:

duração do processo tempo de fim - tempo de início + 1

elapsed time tempo de fim - tempo de chegada + 1

Acompanhe no exemplo feito, usando o método **Round Robin, com q=7**

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
O	2	9	2	16	15	15
I	4	4	9	12	4	9
Y	8	2	13	14	2	7
P	9	7	17	23	7	15
C	14	2	24	25	2	12
U	16	8	26	53	28	38
D	20	5	33	37	5	18
X	24	4	38	41	4	18
J	28	5	42	46	5	19
L	31	6	47	52	6	22
A	33	6	54	59	6	27
M	38	9	60	88	29	51
B	40	6	67	72	6	33
F	42	9	73	90	18	49
N	43	8	80	91	12	49

Veja o resultado do mapa do processador, onde cada letra corresponde a 1 seg. Espaços são só para clareza e separam 10 segundos.

```
 OOOOOOOII IIYYOOPPPP PPPCCUUUUU UUDDDDDXXX XJJJJ
JLLLL LLUAAAAAAM MMMMMMBBBB BBFFFFFFFN NNNNNNMMFF
 N
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 9.9 segundos e 25.5 segundos respectivamente.

☞ Para você fazer

Imagine a seguinte situação em um ambiente multiprogramado com o método **Round Robin, com q=7** .

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
E	1	7				
F	5	3				
K	6	2				
P	7	5				
G	9	9				
W	10	5				
D	12	6				
N	14	5				
B	18	8				
C	25	6				
Z	29	3				
A	32	5				
S	35	2				
U	37	4				
I	45	4				

Para preencher a tabela acima, use o seguinte espaço de tempo

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
100	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150

Responda aqui, qual a média de duração e de elapsed do exercício proposto com o método **Round Robin, com q=7** . (1 casa decimal, por favor)

duração	elapsed

Executando programas

Um programa é uma unidade de trabalho sendo executado em um computador. Em geral, um programa obtém dados de entrada, processa-os e gera dados de saída. Um detalhe importante é que (graças à arquitetura de Von Neumann) programas são tratados e armazenados na memória como se fossem dados.

A CPU utiliza ciclos de máquina repetitivos para executar as instruções do programa, uma a uma, do início ao fim. Cada ciclo é composto pelas fases de busca, decodificação e execução. Na fase de busca, a unidade de controle manda o sistema copiar no registrador de instruções a próxima instrução. (isto é feito a partir do registrador chamado contador do programa). Ao acabar esta fase o contador é incrementado para apontar para a próxima instrução. A fase de decodificação visa preparar os circuitos necessários para obedecer à instrução, seja ela qual for. Finalmente, a execução, é a implementação da instrução.

Note que as velocidades da CPU e dos dispositivos de E/S são completamente díspares, o que implica em algum tipo de sincronização (e obviamente abre as portas para inúmeros níveis de multiprocessamento). Três métodos são usados para obter sincronização: E/S programada (o mais primitivo: a CPU espera pelo dispositivo de E/S), E/S dirigido por interrupção, onde a CPU comanda a transferência, mas não fica aguardando sua conclusão. Aqui o dispositivo de E/S interrompe a CPU quando acabar. Durante este tempo, a CPU pode realizar outras tarefas. Finalmente, o método de Acesso Direto à memória (DMA), transfere um grande bloco de dados entre o dispositivo (rápido) de E/S e a memória, sem precisar passar pela CPU. Isto requer um controlador de DMA.

CPUS modernas

A seguir algumas descrições de processadores modernos, no estado da arte em 2024:

- CPU Intel Core i9-12900K: 5,5 GHz, 16 núcleos, 24 threads, até 5,5 GHz de clock boost, taxa de ciclo de máquina estimada entre 220 e 275 ciclos por instrução (CPI).
- CPU AMD Ryzen 9 5950X: 4,9 GHz, 16 núcleos, 32 threads, até 5,0 GHz de clock boost, taxa de ciclo de máquina estimada entre 200 e 250 CPI.
- CPU Apple M1 Pro: 3,2 GHz, 10 núcleos (8 núcleos de alto desempenho + 2 núcleos de alta eficiência), 16 threads, até 3,2 GHz de clock boost, taxa de ciclo de máquina estimada entre 150 e 200 CPI.

A taxa de ciclo é uma medida da eficiência ao executar uma instrução: ela indica o número de ciclos de relógio necessários para executar 1 instrução. O clock boost é uma execução acelerada que pode ser invocada pela CPU (depende de temperatura, carga, eficiência energética, etc)

Um núcleo é capaz de executar uma instrução por vez. Cada núcleo possui seus próprios recursos dedicados, como registradores e unidades de execução, permitindo que ele processe tarefas de forma independente. Quanto mais núcleos uma CPU possui ela pode lidar com mais tarefas simultaneamente.

As threads, por outro lado, dividem o tempo de processamento com os núcleos. Elas compartilham os mesmos recursos do núcleo, mas podem alternar a execução de diferentes instruções. Isso permite que um único núcleo execute várias threads ao mesmo tempo, aumentando ainda mais a capacidade multitarefa da CPU.

Uma analogia: Imagine uma cozinha como a CPU e os chefs como os núcleos. Cada chef (núcleo) tem sua própria estação de trabalho (recursos dedicados) e pode preparar um prato (instrução) por vez. As garçonetes (threads) circulam pela cozinha, pegando os pratos prontos dos chefs e entregando-os aos clientes (aplicativos). Quanto mais chefs (núcleos) e garçonetes (threads) a cozinha tiver, mais pratos (tarefas) ela poderá preparar e servir (executar) ao mesmo tempo.

Diferentes arquiteturas

CISC acrônimo de *Complex Instruction Set Computer*. Tem um grande conjunto de instruções, incluindo muitas complexas. Torna o processo de programação mais fácil, já que para muitas tarefas, há uma instrução nativa. A complexidade das instruções torna os circuitos da CPU e da UC mais complicados. A maneira de lidar com isso foi construir a programação em dois níveis. No primeiro, chamado micro-operações, estão apenas as operações muito simples, as únicas efetuadas diretamente pela CPU. Surge uma micromemória, que contém a tradução das micro-operações da operação complexa que está sendo executada. Como tudo na vida, há vantagens (programas menores no nível da máquina) e desvantagens (sobrecarga da micromemória e da micro-programação). Um exemplo desta arquitetura é a série Pentium da Intel, bem como mainframes e supercomputadores. Note-se que os PCs parecem estar migrando para a arquitetura RISC.

RISC Acrônimo de *Reduced Instruction Set Computer*. A idéia é ter o mínimo possível no conjunto de instruções de máquina, traduzindo operações complexas em macros usando operações simples sob controle do programa (e não da máquina como no caso CISC). Exemplos de computadores RISC: smartphones e tablets (processadores ARM), consoles de videogame (Nintendo Switch), Apple MacBook Air, entre outros.
Para encerrar vale dizer que mais recentemente computadores começaram a usar recursos de ambas as abordagens.

Pipeline Como se viu, um computador executa operações em 3 fases: busca, decodificação e execução. Originalmente, as 3 fases ocorriam em série para cada instrução: a instrução x precisava concluir as 3 fases antes de começar o ciclo de instrução $x+1$. Mais modernamente os computadores começaram a usar a técnica de *pipeline*, permitindo que a UC possa começar as fases da $x+1$ enquanto ainda trata a x.

Processamento paralelo Quando o custo do hardware começou a cair, computadores começaram a ser desenhados com várias UCs, várias ULAs e várias memórias. Isto permite o paralelismo na execução de instruções. Agora surgem 4 possibilidades: SISD (fluxo único de instruções e único de dados), SIMD (fluxo único de instruções e múltiplo de dados), MISD (único de dados e múltiplo de instruções) e MIMD (múltiplos ambos). Vale notar que o MISD vale como conceito, mas parece não ter sido nunca implementado. A idéia do processamento paralelo, parece mais fácil como conceito do que na prática. Há aqui um problema de balanceamento de carga: a alocação dinâmica de tarefas para que estas sejam executadas nos diversos processardores disponíveis, mas de forma que todos os processadores sejam utilizados adequadamente. Associado a isto está o problema do escalonamento, em inglês *scaling* (divisão de uma tarefa em sub-tarefas de acordo com o número de processadores disponíveis).

Com o aumento do número de tarefas, cresce exponencialmente o trabalho necessário para coordenar a interação entre elas. Se houver 4 tarefas, há potencialmente 6 pares de tarefas se comunicando. Em 5 tarefas, há 10 pares e em 6 tarefas, 15 pares.

Filas dentro do Sist. Operacional

Imagine-se um sistema operacional multiprogramado despachando tarefas. Como o processador é um só, e existem muitos processos querendo-o usar, formam-se filas. Alem do algoritmo FIFO também conhecido como FCFS (*fist come, first served*), vão ser usados 2 algoritmos distintos:

SJF *Shortest Job First.* Neste esquema, quando o processador vai atender alguém, ele escolhe na fila de espera, aquele que tiver a menor requisição de tempo. Se houver empate, o primeiro será retirado.

RR *Round Robin.* Neste esquema preemptivo, a fila será atendida em ordem, mas com uma quantidade fixa de tempo para cada processo. Após receber este *quantum*, o processo vai para o fim da fila.

Para este problema, haverá um padrão de chegadas de requisições de tempo. A unidade a ser considerada é o segundo. Não haverá nenhum tempo de *overhead* para troca de processos. Irão ser usados para cada processo, dois tempos, assim definidos:

duração do processo tempo de fim - tempo de início + 1

elapsed time tempo de fim - tempo de chegada + 1

Acompanhe no exemplo feito, usando o método **Round Robin, com q=7**

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
C	1	6	1	6	6	6
M	4	4	7	10	4	7
W	8	6	11	16	6	9
T	15	11	17	47	31	33
U	16	2	24	25	2	10
R	18	4	26	29	4	12
Z	19	11	30	65	36	47
H	23	11	37	92	56	70
B	27	10	48	100	53	74
D	31	8	55	101	47	71
O	40	9	66	103	38	64
S	41	6	73	78	6	38
K	42	3	79	81	3	40
I	43	7	82	88	7	46
Q	45	5	93	97	5	53

Veja o resultado do mapa do processador, onde cada letra corresponde a 1 seg. Espaços são só para clareza e separam 10 segundos.

```
CCCCCCMMMM WWWWWWTTTT TTTUURRRRZ ZZZZZZHHHH HHHTT
TTBBB BBBBDDDDDD DZZZZOOOOO OOSSSSSSKK KIIIIIIIHH
 HHQQQQQBBB DOO
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 20.3 segundos e 38.7 segundos respectivamente.

☞ Para você fazer

Imagine a seguinte situação em um ambiente multiprogramado com o método **Round Robin, com q=7** .

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
O	1	9				
W	2	4				
K	7	5				
S	8	11				
H	16	2				
Y	17	11				
F	19	9				
X	20	8				
J	26	5				
N	31	8				
D	32	3				
L	34	5				
R	35	7				
U	36	5				
E	41	9				

Para preencher a tabela acima, use o seguinte espaço de tempo

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
100	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150

Responda aqui, qual a média de duração e de elapsed do exercício proposto com o método **Round Robin, com q=7** . (1 casa decimal, por favor)

duração	elapsed

CEP - UP - UTFPR - PUCPr - UFPR – 03/06/2024 - 13:31:33.3
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V: 2.02 76537 FELIPE EIJI KANO 24CC2204 - 5 / _____ / _____

Executando programas

Um programa é uma unidade de trabalho sendo executado em um computador. Em geral, um programa obtém dados de entrada, processa-os e gera dados de saída. Um detalhe importante é que (graças à arquitetura de Von Neumann) programas são tratados e armazenados na memória como se fossem dados.

A CPU utiliza ciclos de máquina repetitivos para executar as instruções do programa, uma a uma, do início ao fim. Cada ciclo é composto pelas fases de busca, decodificação e execução. Na fase de busca, a unidade de controle manda o sistema copiar no registrador de instruções a próxima instrução. (isto é feito a partir do registrador chamado contador do programa). Ao acabar esta fase o contador é incrementado para apontar para a próxima instrução. A fase de decodificação visa preparar os circuitos necessários para obedecer à instrução, seja ela qual for. Finalmente, a execução, é a implementação da instrução.

Note que as velocidades da CPU e dos dispositivos de E/S são completamente díspares, o que implica em algum tipo de sincronização (e obviamente abre as portas para inúmeros níveis de multiprocessamento). Três métodos são usados para obter sincronização: E/S programada (o mais primitivo: a CPU espera pelo dispositivo de E/S), E/S dirigido por interrupção, onde a CPU comanda a transferência, mas não fica aguardando sua conclusão. Aqui o dispositivo de E/S interrompe a CPU quando acabar. Durante este tempo, a CPU pode realizar outras tarefas. Finalmente, o método de Acesso Direto à memória (DMA), transfere um grande bloco de dados entre o dispositivo (rápido) de E/S e a memória, sem precisar passar pela CPU. Isto requer um controlador de DMA.

CPUS modernas
A seguir algumas descrições de processadores modernos, no estado da arte em 2024:

- CPU Intel Core i9-12900K: 5,5 GHz, 16 núcleos, 24 threads, até 5,5 GHz de clock boost, taxa de ciclo de máquina estimada entre 220 e 275 ciclos por instrução (CPI).
- CPU AMD Ryzen 9 5950X: 4,9 GHz, 16 núcleos, 32 threads, até 5,0 GHz de clock boost, taxa de ciclo de máquina estimada entre 200 e 250 CPI.
- CPU Apple M1 Pro: 3,2 GHz, 10 núcleos (8 núcleos de alto desempenho + 2 núcleos de alta eficiência), 16 threads, até 3,2 GHz de clock boost, taxa de ciclo de máquina estimada entre 150 e 200 CPI.

A taxa de ciclo é uma medida da eficiência ao executar uma instrução: ela indica o número de ciclos de relógio necessários para executar 1 instrução. O clock boost é uma execução acelerada que pode ser invocada pela CPU (depende de temperatura, carga, eficiência energética, etc)

Um núcleo é capaz de executar uma instrução por vez. Cada núcleo possui seus próprios recursos dedicados, como registradores e unidades de execução, permitindo que ele processe tarefas de forma independente. Quanto mais núcleos uma CPU possui ela pode lidar com mais tarefas simultaneamente.

As threads, por outro lado, dividem o tempo de processamento com os núcleos. Elas compartilham os mesmos recursos do núcleo, mas podem alternar a execução de diferentes instruções. Isso permite que um único núcleo execute várias threads ao mesmo tempo, aumentando ainda mais a capacidade multitarefa da CPU.

Uma analogia: Imagine uma cozinha como a CPU e os chefs como os núcleos. Cada chef (núcleo) tem sua própria estação de trabalho (recursos dedicados) e pode preparar um prato (instrução) por vez. As garçonetes (threads) circulam pela cozinha, pegando os pratos prontos dos chefs e entregando-os aos clientes (aplicativos). Quanto mais chefs (núcleos) e garçonetes (threads) a cozinha tiver, mais pratos (tarefas) ela poderá preparar e servir (executar) ao mesmo tempo.

Diferentes arquiteturas

CISC acrônimo de *Complex Instruction Set Computer*. Tem um grande conjunto de instruções, incluindo muitas complexas. Torna o processo de programação mais fácil, já que para muitas tarefas, há uma instrução nativa. A complexidade das instruções torna os circuitos da CPU e da UC mais complicados. A maneira de lidar com isso foi construir a programação em dois níveis. No primeiro, chamado micro-operações, estão apenas as operações muito simples, as únicas efetuadas diretamente pela CPU. Surge uma micromemória, que contém a tradução das micro-operações da operação complexa que está sendo executada. Como tudo na vida, há vantagens (programas menores no nível da máquina) e desvantagens (sobrecarga da micromemória e da micro-programação). Um exemplo desta arquitetura é a série Pentium da Intel, bem como mainframes e supercomputadores. Note-se que os PCs parecem estar migrando para a arquitetura RISC.

RISC Acrônimo de *Reduced Instruction Set Computer*. A idéia é ter o mínimo possível no conjunto de instruções de máquina, traduzindo operações complexas em macros usando operações simples sob controle do programa (e não da máquina como no caso CISC). Exemplos de computadores RISC: smartphones e tablets (processadores ARM), consoles de videogame (Nintendo Switch), Apple MacBook Air, entre outros.
Para encerrar vale dizer que mais recentemente computadores começaram a usar recursos de ambas as abordagens.

Pipeline Como se viu, um computador executa operações em 3 fases: busca, decodificação e execução. Originalmente, as 3 fases ocorriam em série para cada instrução: a instrução x precisava concluir as 3 fases antes de começar o ciclo de instrução $x+1$. Mais modernamente os computadores começaram a usar a técnica de *pipeline*, permitindo que a UC possa começar as fases da $x+1$ enquanto ainda trata a x.

Processamento paralelo Quando o custo do hardware começou a cair, computadores começaram a ser desenhados com várias UCs, várias ULAs e várias memórias. Isto permite o paralelismo na execução de instruções. Agora surgem 4 possibilidades: SISD (fluxo único de instruções e único de dados), SIMD (fluxo único de instruções e múltiplo de dados), MISD (único de dados e múltiplo de instruções) e MIMD (múltiplos ambos). Vale notar que o MISD vale como conceito, mas parece não ter sido nunca implementado. A idéia do processamento paralelo, parece mais fácil como conceito do que na prática. Há aqui um problema de balanceamento de carga: a alocação dinâmica de tarefas para que estas sejam executadas nos diversos processardores disponíveis, mas de forma que todos os processadores sejam utilizados adequadamente. Associado a isto está o problema do escalonamento, em inglês *scaling* (divisão de uma tarefa em sub-tarefas de acordo com o número de processadores disponíveis).

Com o aumento do número de tarefas, cresce exponencialmente o trabalho necessário para coordenar a interação entre elas. Se houver 4 tarefas, há potencialmente 6 pares de tarefas se comunicando. Em 5 tarefas, há 10 pares e em 6 tarefas, 15 pares.

Filas dentro do Sist. Operacional

Imagine-se um sistema operacional multiprogramado despachando tarefas. Como o processador é um só, e existem muitos processos querendo-o usar, formam-se filas. Alem do algoritmo FIFO também conhecido como FCFS (*fist come, first served*), vão ser usados 2 algoritmos distintos:

SJF *Shortest Job First.* Neste esquema, quando o processador vai atender alguém, ele escolhe na fila de espera, aquele que tiver a menor requisição de tempo. Se houver empate, o primeiro será retirado.

RR *Round Robin.* Neste esquema preemptivo, a fila será atendida em ordem, mas com uma quantidade fixa de tempo para cada processo. Após receber este *quantum*, o processo vai para o fim da fila.

Para este problema, haverá um padrão de chegadas de requisições de tempo. A unidade a ser considerada é o segundo. Não haverá nenhum tempo de *overhead* para troca de processos. Irão ser usados para cada processo, dois tempos, assim definidos:

duração do processo tempo de fim - tempo de início + 1

elapsed time tempo de fim - tempo de chegada + 1

Acompanhe no exemplo feito, usando o método **Round Robin, com q=3**

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
P	1	10	1	10	10	10
J	11	11	11	24	14	14
G	18	7	20	52	33	35
K	20	4	25	42	18	23
Q	22	11	28	88	61	67
X	23	2	34	35	2	13
A	24	3	36	38	3	15
Y	25	4	39	59	21	35
L	29	6	43	71	29	43
I	33	11	49	96	48	64
U	35	3	53	55	3	21
H	40	7	56	92	37	53
O	42	7	60	93	34	52
T	43	7	63	94	32	52
W	44	3	66	68	3	25

Veja o resultado do mapa do processador, onde cada letra corresponde a 1 seg. Espaços são só para clareza e separam 10 segundos.

```
PPPPPPPPPP JJJJJJJJJG GGJJKKKQQQ GGGXXAAAYY YKLLL
QQQII IGUUUHHHYO OOTTTWWWLL LQQQIIIHHH OOOTTTQQII
 IHOTII
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 23.2 segundos e 34.8 segundos respectivamente.

☞ Para você fazer

Imagine a seguinte situação em um ambiente multiprogramado com o método **Round Robin, com q=3** .

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
B	1	10				
T	3	2				
V	7	4				
Z	9	8				
F	10	2				
U	19	6				
Y	20	3				
Q	30	3				
H	31	5				
S	33	7				
G	36	9				
O	38	11				
D	41	9				
P	42	11				
E	43	8				

Para preencher a tabela acima, use o seguinte espaço de tempo

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
100	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150

Responda aqui, qual a média de duração e de elapsed do exercício proposto com o método **Round Robin, com q=3** . (1 casa decimal, por favor)

duração	elapsed

204-76537 - /

CEP - UP - UTFPR - PUCPr - UFPR – 03/06/2024 - 13:31:33.3
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V: 2.02 76544 FELIPE JESMIEL LEITE 24CC2204 - 6 _____ / _____ / _____

Executando programas

Um programa é uma unidade de trabalho sendo executado em um computador. Em geral, um programa obtém dados de entrada, processa-os e gera dados de saída. Um detalhe importante é que (graças à arquitetura de Von Neumann) programas são tratados e armazenados na memória como se fossem dados.

A CPU utiliza ciclos de máquina repetitivos para executar as instruções do programa, uma a uma, do início ao fim. Cada ciclo é composto pelas fases de busca, decodificação e execução. Na fase de busca, a unidade de controle manda o sistema copiar no registrador de instruções a próxima instrução. (isto é feito a partir do registrador chamado contador do programa). Ao acabar esta fase o contador é incrementado para apontar para a próxima instrução. A fase de decodificação visa preparar os circuitos necessários para obedecer à instrução, seja ela qual for. Finalmente, a execução, é a implementação da instrução.

Note que as velocidades da CPU e dos dispositivos de E/S são completamente díspares, o que implica em algum tipo de sincronização (e obviamente abre as portas para inúmeros níveis de multiprocessamento). Três métodos são usados para obter sincronização: E/S programada (o mais primitivo: a CPU espera pelo dispositivo de E/S), E/S dirigido por interrupção, onde a CPU comanda a transferência, mas não fica aguardando sua conclusão. Aqui o dispositivo de E/S interrompe a CPU quando acabar. Durante este tempo, a CPU pode realizar outras tarefas. Finalmente, o método de Acesso Direto à memória (DMA), transfere um grande bloco de dados entre o dispositivo (rápido) de E/S e a memória, sem precisar passar pela CPU. Isto requer um controlador de DMA.

CPUS modernas
A seguir algumas descrições de processadores modernos, no estado da arte em 2024:

- CPU Intel Core i9-12900K: 5,5 GHz, 16 núcleos, 24 threads, até 5,5 GHz de clock boost, taxa de ciclo de máquina estimada entre 220 e 275 ciclos por instrução (CPI).
- CPU AMD Ryzen 9 5950X: 4,9 GHz, 16 núcleos, 32 threads, até 5,0 GHz de clock boost, taxa de ciclo de máquina estimada entre 200 e 250 CPI.
- CPU Apple M1 Pro: 3,2 GHz, 10 núcleos (8 núcleos de alto desempenho + 2 núcleos de alta eficiência), 16 threads, até 3,2 GHz de clock boost, taxa de ciclo de máquina estimada entre 150 e 200 CPI.

A taxa de ciclo é uma medida da eficiência ao executar uma instrução: ela indica o número de ciclos de relógio necessários para executar 1 instrução. O clock boost é uma execução acelerada que pode ser invocada pela CPU (depende de temperatura, carga, eficiência energética, etc)

Um núcleo é capaz de executar uma instrução por vez. Cada núcleo possui seus próprios recursos dedicados, como registradores e unidades de execução, permitindo que ele processe tarefas de forma independente. Quanto mais núcleos uma CPU possui ela pode lidar com mais tarefas simultaneamente.

As threads, por outro lado, dividem o tempo de processamento com os núcleos. Elas compartilham os mesmos recursos do núcleo, mas podem alternar a execução de diferentes instruções. Isso permite que um único núcleo execute várias threads ao mesmo tempo, aumentando ainda mais a capacidade multitarefa da CPU.

Uma analogia: Imagine uma cozinha como a CPU e os chefs como os núcleos. Cada chef (núcleo) tem sua própria estação de trabalho (recursos dedicados) e pode preparar um prato (instrução) por vez. As garçonetes (threads) circulam pela cozinha, pegando os pratos prontos dos chefs e entregando-os aos clientes (aplicativos). Quanto mais chefs (núcleos) e garçonetes (threads) a cozinha tiver, mais pratos (tarefas) ela poderá preparar e servir (executar) ao mesmo tempo.

Diferentes arquiteturas

CISC acrônimo de *Complex Instruction Set Computer*. Tem um grande conjunto de instruções, incluindo muitas complexas. Torna o processo de programação mais fácil, já que para muitas tarefas, há uma instrução nativa. A complexidade das instruções torna os circuitos da CPU e da UC mais complicados. A maneira de lidar com isso foi construir a programação em dois níveis. No primeiro, chamado micro-operações, estão apenas as operações muito simples, as únicas efetuadas diretamente pela CPU. Surge uma micromemória, que contém a tradução das micro-operações da operação complexa que está sendo executada. Como tudo na vida, há vantagens (programas menores no nível da máquina) e desvantagens (sobrecarga da micromemória e da micro-programação). Um exemplo desta arquitetura é a série Pentium da Intel, bem como mainframes e supercomputadores. Note-se que os PCs parecem estar migrando para a arquitetura RISC.

RISC Acrônimo de *Reduced Instruction Set Computer*. A idéia é ter o mínimo possível no conjunto de instruções de máquina, traduzindo operações complexas em macros usando operações simples sob controle do programa (e não da máquina como no caso CISC). Exemplos de computadores RISC: smartphones e tablets (processadores ARM), consoles de videogame (Nintendo Switch), Apple MacBook Air, entre outros.
Para encerrar vale dizer que mais recentemente computadores começaram a usar recursos de ambas as abordagens.

Pipeline Como se viu, um computador executa operações em 3 fases: busca, decodificação e execução. Originalmente, as 3 fases ocorriam em série para cada instrução: a instrução x precisava concluir as 3 fases antes de começar o ciclo de instrução $x+1$. Mais modernamente os computadores começaram a usar a técnica de *pipeline*, permitindo que a UC possa começar as fases da $x+1$ enquanto ainda trata a x.

Processamento paralelo Quando o custo do hardware começou a cair, computadores começaram a ser desenhados com várias UCs, várias ULAs e várias memórias. Isto permite o paralelismo na execução de instruções. Agora surgem 4 possibilidades: SISD (fluxo único de instruções e único de dados), SIMD (fluxo único de instruções e múltiplo de dados), MISD (único de dados e múltiplo de instruções) e MIMD (múltiplos ambos). Vale notar que o MISD vale como conceito, mas parece não ter sido nunca implementado. A idéia do processamento paralelo, parece mais fácil como conceito do que na prática. Há aqui um problema de balanceamento de carga: a alocação dinâmica de tarefas para que estas sejam executadas nos diversos processardores disponíveis, mas de forma que todos os processadores sejam utilizados adequadamente. Associado a isto está o problema do escalonamento, em inglês *scaling* (divisão de uma tarefa em sub-tarefas de acordo com o número de processadores disponíveis).

Com o aumento do número de tarefas, cresce exponencialmente o trabalho necessário para coordenar a interação entre elas. Se houver 4 tarefas, há potencialmente 6 pares de tarefas se comunicando. Em 5 tarefas, há 10 pares e em 6 tarefas, 15 pares.

Filas dentro do Sist. Operacional

Imagine-se um sistema operacional multiprogramado despachando tarefas. Como o processador é um só, e existem muitos processos querendo-o usar, formam-se filas. Alem do algoritmo FIFO também conhecido como FCFS (*fist come, first served*), vão ser usados 2 algoritmos distintos:

SJF *Shortest Job First*. Neste esquema, quando o processador vai atender alguém, ele escolhe na fila de espera, aquele que tiver a menor requisição de tempo. Se houver empate, o primeiro será retirado.

RR *Round Robin*. Neste esquema preemptivo, a fila será atendida em ordem, mas com uma quantidade fixa de tempo para cada processo. Após receber este *quantum*, o processo vai para o fim da fila.

Para este problema, haverá um padrão de chegadas de requisições de tempo. A unidade a ser considerada é o segundo. Não haverá nenhum tempo de *overhead* para troca de processos. Irão ser usados para cada processo, dois tempos, assim definidos:

duração do processo tempo de fim - tempo de início + 1

elapsed time tempo de fim - tempo de chegada + 1

Acompanhe no exemplo feito, usando o método **Shortest Job First**

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
L	2	8	2	9	8	8
P	8	11	20	30	11	23
D	10	6	10	15	6	6
R	11	2	16	17	2	7
E	14	2	18	19	2	6
U	22	7	52	58	7	37
Q	25	6	46	51	6	27
Z	28	4	31	34	4	7
F	29	10	75	84	10	56
V	30	4	35	38	4	9
I	31	9	66	74	9	44
S	35	7	59	65	7	31
N	38	4	39	42	4	5
Y	43	3	43	45	3	3
C	44	11	85	95	11	52

Veja o resultado do mapa do processador, onde cada letra corresponde a 1 seg. Espaços são só para clareza e separam 10 segundos.

```
LLLLLLLLD DDDDDRREEP PPPPPPPPPP ZZZZVVVVNN NNYYY
QQQQQ QUUUUUUUSS SSSSSIIIII IIIIFFFFFF FFFFCCCCCC
 CCCCC
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 6.3 segundos e 21.4 segundos respectivamente.

☞ Para você fazer

Imagine a seguinte situação em um ambiente multiprogramado com o método **Shortest Job First** .

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
N	8	8				
J	9	2				
P	11	7				
W	15	10				
V	19	6				
R	25	4				
C	26	2				
Y	29	2				
S	30	10				
H	32	3				
A	34	7				
G	37	5				
U	42	2				
T	44	9				
Z	45	11				

Para preencher a tabela acima, use o seguinte espaço de tempo

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
100	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150

Responda aqui, qual a média de duração e de elapsed do exercício proposto com o método **Shortest Job First** . (1 casa decimal, por favor)

duração	elapsed

204-76544 - /

CEP - UP - UTFPR - PUCPr - UFPR –
03/06/2024 - 13:31:33.3
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V: 2.02 76551 GABRIELA BERNARDO PASQUALOTTO 24CC2204 - 7 _____ / _____ /

Executando programas

Um programa é uma unidade de trabalho sendo executado em um computador. Em geral, um programa obtém dados de entrada, processa-os e gera dados de saída. Um detalhe importante é que (graças à arquitetura de Von Neumann) programas são tratados e armazenados na memória como se fossem dados.

A CPU utiliza ciclos de máquina repetitivos para executar as instruções do programa, uma a uma, do início ao fim. Cada ciclo é composto pelas fases de busca, decodificação e execução. Na fase de busca, a unidade de controle manda o sistema copiar no registrador de instruções a próxima instrução. (isto é feito a partir do registrador chamado contador do programa). Ao acabar esta fase o contador é incrementado para apontar para a próxima instrução. A fase de decodificação visa preparar os circuitos necessários para obedecer à instrução, seja ela qual for. Finalmente, a execução, é a implementação da instrução.

Note que as velocidades da CPU e dos dispositivos de E/S são completamente díspares, o que implica em algum tipo de sincronização (e obviamente abre as portas para inúmeros níveis de multiprocessamento). Três métodos são usados para obter sincronização: E/S programada (o mais primitivo: a CPU espera pelo dispositivo de E/S), E/S dirigido por interrupção, onde a CPU comanda a transferência, mas não fica aguardando sua conclusão. Aqui o dispositivo de E/S interrompe a CPU quando acabar. Durante este tempo, a CPU pode realizar outras tarefas. Finalmente, o método de Acesso Direto à memória (DMA), transfere um grande bloco de dados entre o dispositivo (rápido) de E/S e a memória, sem precisar passar pela CPU. Isto requer um controlador de DMA.

CPUS modernas

A seguir algumas descrições de processadores modernos, no estado da arte em 2024:

- CPU Intel Core i9-12900K: 5,5 GHz, 16 núcleos, 24 threads, até 5,5 GHz de clock boost, taxa de ciclo de máquina estimada entre 220 e 275 ciclos por instrução (CPI).
- CPU AMD Ryzen 9 5950X: 4,9 GHz, 16 núcleos, 32 threads, até 5,0 GHz de clock boost, taxa de ciclo de máquina estimada entre 200 e 250 CPI.
- CPU Apple M1 Pro: 3,2 GHz, 10 núcleos (8 núcleos de alto desempenho + 2 núcleos de alta eficiência), 16 threads, até 3,2 GHz de clock boost, taxa de ciclo de máquina estimada entre 150 e 200 CPI.

A taxa de ciclo é uma medida da eficiência ao executar uma instrução: ela indica o número de ciclos de relógio necessários para executar 1 instrução. O clock boost é uma execução acelerada que pode ser invocada pela CPU (depende de temperatura, carga, eficiência energética, etc)

Um núcleo é capaz de executar uma instrução por vez. Cada núcleo possui seus próprios recursos dedicados, como registradores e unidades de execução, permitindo que ele processe tarefas de forma independente. Quanto mais núcleos uma CPU possui ela pode lidar com mais tarefas simultaneamente.

As threads, por outro lado, dividem o tempo de processamento com os núcleos. Elas compartilham os mesmos recursos do núcleo, mas podem alternar a execução de diferentes instruções. Isso permite que um único núcleo execute várias threads ao mesmo tempo, aumentando ainda mais a capacidade multitarefa da CPU.

Uma analogia: Imagine uma cozinha como a CPU e os chefs como os núcleos. Cada chef (núcleo) tem sua própria estação de trabalho (recursos dedicados) e pode preparar um prato (instrução) por vez. As garçonetes (threads) circulam pela cozinha, pegando os pratos prontos dos chefs e entregando-os aos clientes (aplicativos). Quanto mais chefs (núcleos) e garçonetes (threads) a cozinha tiver, mais pratos (tarefas) ela poderá preparar e servir (executar) ao mesmo tempo.

Diferentes arquiteturas

CISC acrônimo de *Complex Instruction Set Computer*. Tem um grande conjunto de instruções, incluindo muitas complexas. Torna o processo de programação mais fácil, já que para muitas tarefas, há uma instrução nativa. A complexidade das instruções torna os circuitos da CPU e da UC mais complicados. A maneira de lidar com isso foi construir a programação em dois níveis. No primeiro, chamado micro-operações, estão apenas as operações muito simples, as únicas efetuadas diretamente pela CPU. Surge uma micromemória, que contém a tradução das micro-operações da operação complexa que está sendo executada. Como tudo na vida, há vantagens (programas menores no nível da máquina) e desvantagens (sobrecarga da micromemória e da micro-programação). Um exemplo desta arquitetura é a série Pentium da Intel, bem como mainframes e supercomputadores. Note-se que os PCs parecem estar migrando para a arquitetura RISC.

RISC Acrônimo de *Reduced Instruction Set Computer*. A idéia é ter o mínimo possível no conjunto de instruções de máquina, traduzindo operações complexas em macros usando operações simples sob controle do programa (e não da máquina como no caso CISC). Exemplos de computadores RISC: smartphones e tablets (processadores ARM), consoles de videogame (Nintendo Switch), Apple MacBook Air, entre outros.
Para encerrar vale dizer que mais recentemente computadores começaram a usar recursos de ambas as abordagens.

Pipeline Como se viu, um computador executa operações em 3 fases: busca, decodificação e execução. Originalmente, as 3 fases ocorriam em série para cada instrução: a instrução x precisava concluir as 3 fases antes de começar o ciclo de instrução $x+1$. Mais modernamente os computadores começaram a usar a técnica de *pipeline*, permitindo que a UC possa começar as fases da $x+1$ enquanto ainda trata a x.

Processamento paralelo Quando o custo do hardware começou a cair, computadores começaram a ser desenhados com várias UCs, várias ULAs e várias memórias. Isto permite o paralelismo na execução de instruções. Agora surgem 4 possibilidades: SISD (fluxo único de instruções e único de dados), SIMD (fluxo único de instruções e múltiplo de dados), MISD (único de dados e múltiplo de instruções) e MIMD (múltiplos ambos). Vale notar que o MISD vale como conceito, mas parece não ter sido nunca implementado. A idéia do processamento paralelo, parece mais fácil como conceito do que na prática. Há aqui um problema de balanceamento de carga: a alocação dinâmica de tarefas para que estas sejam executadas nos diversos processardores disponíveis, mas de forma que todos os processadores sejam utilizados adequadamente. Associado a isto está o problema do escalonamento, em inglês *scaling* (divisão de uma tarefa em sub-tarefas de acordo com o número de processadores disponíveis).

Com o aumento do número de tarefas, cresce exponencialmente o trabalho necessário para coordenar a interação entre elas. Se houver 4 tarefas, há potencialmente 6 pares de tarefas se comunicando. Em 5 tarefas, há 10 pares e em 6 tarefas, 15 pares.

Filas dentro do Sist. Operacional

Imagine-se um sistema operacional multiprogramado despachando tarefas. Como o processador é um só, e existem muitos processos querendo-o usar, formam-se filas. Alem do algoritmo FIFO também conhecido como FCFS (*fist come, first served*), vão ser usados 2 algoritmos distintos:

SJF *Shortest Job First.* Neste esquema, quando o processador vai atender alguém, ele escolhe na fila de espera, aquele que tiver a menor requisição de tempo. Se houver empate, o primeiro será retirado.

RR *Round Robin.* Neste esquema preemptivo, a fila será atendida em ordem, mas com uma quantidade fixa de tempo para cada processo. Após receber este *quantum*, o processo vai para o fim da fila.

Para este problema, haverá um padrão de chegadas de requisições de tempo. A unidade a ser considerada é o segundo. Não haverá nenhum tempo de *overhead* para troca de processos. Irão ser usados para cada processo, dois tempos, assim definidos:

duração do processo tempo de fim - tempo de início + 1

elapsed time tempo de fim - tempo de chegada + 1

Acompanhe no exemplo feito, usando o método **Round Robin, com q=7**

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
I	2	3	2	4	3	3
Z	4	2	5	6	2	3
P	6	6	7	12	6	7
K	7	6	13	18	6	12
U	9	7	19	25	7	17
F	14	3	26	28	3	15
X	16	2	29	30	2	15
Q	21	11	31	65	35	45
J	25	9	38	88	51	64
E	27	4	45	48	4	22
H	31	6	49	54	6	24
T	34	11	55	92	38	59
W	40	10	66	95	30	56
R	41	7	73	79	7	39
A	44	10	80	98	19	55

Veja o resultado do mapa do processador, onde cada letra corresponde a 1 seg. Espaços são só para clareza e separam 10 segundos.

```
 IIIZZPPPP PPKKKKKKUU UUUUUFFFXX QQQQQQQJJJ JJJJE
EEEHH HHHHTTTTTT TQQQQWWWWW WWRRRRRRRA AAAAAAJJTT
 TTWWWAAA
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 14.6 segundos e 29.1 segundos respectivamente.

☞ Para você fazer

Imagine a seguinte situação em um ambiente multiprogramado com o método **Round Robin, com q=7** .

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
C	4	11				
I	5	3				
D	6	3				
U	9	8				
Y	10	3				
W	12	4				
Q	13	11				
N	20	10				
S	25	7				
B	27	3				
E	28	3				
T	30	10				
X	32	7				
A	36	11				
K	42	11				

Para preencher a tabela acima, use o seguinte espaço de tempo

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
100	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150

Responda aqui, qual a média de duração e de elapsed do exercício proposto com o método **Round Robin, com q=7** . (1 casa decimal, por favor)

duração	elapsed

CEP - UP - UTFPR - PUCPr - UFPR –
03/06/2024 - 13:31:33.3
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V:
2.02 76568 GUILHERME VITOR DA SILVA
24CC2204 - 8 _____ / _____ / _____

Executando programas

Um programa é uma unidade de trabalho sendo executado em um computador. Em geral, um programa obtém dados de entrada, processa-os e gera dados de saída. Um detalhe importante é que (graças à arquitetura de Von Neumann) programas são tratados e armazenados na memória como se fossem dados.

A CPU utiliza ciclos de máquina repetitivos para executar as instruções do programa, uma a uma, do início ao fim. Cada ciclo é composto pelas fases de busca, decodificação e execução. Na fase de busca, a unidade de controle manda o sistema copiar no registrador de instruções a próxima instrução. (isto é feito a partir do registrador chamado contador do programa). Ao acabar esta fase o contador é incrementado para apontar para a próxima instrução. A fase de decodificação visa preparar os circuitos necessários para obedecer à instrução, seja ela qual for. Finalmente, a execução, é a implementação da instrução.

Note que as velocidades da CPU e dos dispositivos de E/S são completamente díspares, o que implica em algum tipo de sincronização (e obviamente abre as portas para inúmeros níveis de multiprocessamento). Três métodos são usados para obter sincronização: E/S programada (o mais primitivo: a CPU espera pelo dispositivo de E/S), E/S dirigido por interrupção, onde a CPU comanda a transferência, mas não fica aguardando sua conclusão. Aqui o dispositivo de E/S interrompe a CPU quando acabar. Durante este tempo, a CPU pode realizar outras tarefas. Finalmente, o método de Acesso Direto à memória (DMA), transfere um grande bloco de dados entre o dispositivo (rápido) de E/S e a memória, sem precisar passar pela CPU. Isto requer um controlador de DMA.

CPUS modernas
A seguir algumas descrições de processadores modernos, no estado da arte em 2024:

- CPU Intel Core i9-12900K: 5,5 GHz, 16 núcleos, 24 threads, até 5,5 GHz de clock boost, taxa de ciclo de máquina estimada entre 220 e 275 ciclos por instrução (CPI).
- CPU AMD Ryzen 9 5950X: 4,9 GHz, 16 núcleos, 32 threads, até 5,0 GHz de clock boost, taxa de ciclo de máquina estimada entre 200 e 250 CPI.
- CPU Apple M1 Pro: 3,2 GHz, 10 núcleos (8 núcleos de alto desempenho + 2 núcleos de alta eficiência), 16 threads, até 3,2 GHz de clock boost, taxa de ciclo de máquina estimada entre 150 e 200 CPI.

A taxa de ciclo é uma medida da eficiência ao executar uma instrução: ela indica o número de ciclos de relógio necessários para executar 1 instrução. O clock boost é uma execução acelerada que pode ser invocada pela CPU (depende de temperatura, carga, eficiência energética, etc)

Um núcleo é capaz de executar uma instrução por vez. Cada núcleo possui seus próprios recursos dedicados, como registradores e unidades de execução, permitindo que ele processe tarefas de forma independente. Quanto mais núcleos uma CPU possui ela pode lidar com mais tarefas simultaneamente.

As threads, por outro lado, dividem o tempo de processamento com os núcleos. Elas compartilham os mesmos recursos do núcleo, mas podem alternar a execução de diferentes instruções. Isso permite que um único núcleo execute várias threads ao mesmo tempo, aumentando ainda mais a capacidade multitarefa da CPU.

Uma analogia: Imagine uma cozinha como a CPU e os chefs como os núcleos. Cada chef (núcleo) tem sua própria estação de trabalho (recursos dedicados) e pode preparar um prato (instrução) por vez. As garçonetes (threads) circulam pela cozinha, pegando os pratos prontos dos chefs e entregando-os aos clientes (aplicativos). Quanto mais chefs (núcleos) e garçonetes (threads) a cozinha tiver, mais pratos (tarefas) ela poderá preparar e servir (executar) ao mesmo tempo.

Diferentes arquiteturas

CISC acrônimo de *Complex Instruction Set Computer*. Tem um grande conjunto de instruções, incluindo muitas complexas. Torna o processo de programação mais fácil, já que para muitas tarefas, há uma instrução nativa. A complexidade das instruções torna os circuitos da CPU e da UC mais complicados. A maneira de lidar com isso foi construir a programação em dois níveis. No primeiro, chamado micro-operações, estão apenas as operações muito simples, as únicas efetuadas diretamente pela CPU. Surge uma micromemória, que contém a tradução das micro-operações da operação complexa que está sendo executada. Como tudo na vida, há vantagens (programas menores no nível da máquina) e desvantagens (sobrecarga da micromemória e da micro-programação). Um exemplo desta arquitetura é a série Pentium da Intel, bem como mainframes e supercomputadores. Note-se que os PCs parecem estar migrando para a arquitetura RISC.

RISC Acrônimo de *Reduced Instruction Set Computer*. A idéia é ter o mínimo possível no conjunto de instruções de máquina, traduzindo operações complexas em macros usando operações simples sob controle do programa (e não da máquina como no caso CISC). Exemplos de computadores RISC: smartphones e tablets (processadores ARM), consoles de videogame (Nintendo Switch), Apple MacBook Air, entre outros.
Para encerrar vale dizer que mais recentemente computadores começaram a usar recursos de ambas as abordagens.

Pipeline Como se viu, um computador executa operações em 3 fases: busca, decodificação e execução. Originalmente, as 3 fases ocorriam em série para cada instrução: a instrução x precisava concluir as 3 fases antes de começar o ciclo de instrução $x+1$. Mais modernamente os computadores começaram a usar a técnica de *pipeline*, permitindo que a UC possa começar as fases da $x+1$ enquanto ainda trata a x.

Processamento paralelo Quando o custo do hardware começou a cair, computadores começaram a ser desenhados com várias UCs, várias ULAs e várias memórias. Isto permite o paralelismo na execução de instruções. Agora surgem 4 possibilidades: SISD (fluxo único de instruções e único de dados), SIMD (fluxo único de instruções e múltiplo de dados), MISD (único de dados e múltiplo de instruções) e MIMD (múltiplos ambos). Vale notar que o MISD vale como conceito, mas parece não ter sido nunca implementado. A idéia do processamento paralelo, parece mais fácil como conceito do que na prática. Há aqui um problema de balanceamento de carga: a alocação dinâmica de tarefas para que estas sejam executadas nos diversos processardores disponíveis, mas de forma que todos os processadores sejam utilizados adequadamente. Associado a isto está o problema do escalonamento, em inglês *scaling* (divisão de uma tarefa em sub-tarefas de acordo com o número de processadores disponíveis).

Com o aumento do número de tarefas, cresce exponencialmente o trabalho necessário para coordenar a interação entre elas. Se houver 4 tarefas, há potencialmente 6 pares de tarefas se comunicando. Em 5 tarefas, há 10 pares e em 6 tarefas, 15 pares.

Filas dentro do Sist. Operacional

Imagine-se um sistema operacional multiprogramado despachando tarefas. Como o processador é um só, e existem muitos processos querendo-o usar, formam-se filas. Alem do algoritmo FIFO também conhecido como FCFS (*fist come, first served*), vão ser usados 2 algoritmos distintos:

SJF *Shortest Job First.* Neste esquema, quando o processador vai atender alguém, ele escolhe na fila de espera, aquele que tiver a menor requisição de tempo. Se houver empate, o primeiro será retirado.

RR *Round Robin.* Neste esquema preemptivo, a fila será atendida em ordem, mas com uma quantidade fixa de tempo para cada processo. Após receber este *quantum*, o processo vai para o fim da fila.

Para este problema, haverá um padrão de chegadas de requisições de tempo. A unidade a ser considerada é o segundo. Não haverá nenhum tempo de *overhead* para troca de processos. Irão ser usados para cada processo, dois tempos, assim definidos:

duração do processo tempo de fim - tempo de início + 1

elapsed time tempo de fim - tempo de chegada + 1

Acompanhe no exemplo feito, usando o método **Round Robin, com q=3**

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
H	3	2	3	4	2	2
U	4	8	5	12	8	9
M	19	7	19	31	13	13
I	22	5	25	39	15	18
Q	23	6	28	45	18	23
O	26	3	32	34	3	9
R	27	8	35	78	44	52
E	30	3	40	42	3	13
B	31	4	46	70	25	40
Y	33	10	49	99	51	67
J	36	7	52	91	40	56
P	38	11	58	101	44	64
S	41	11	61	103	43	63
N	42	3	64	66	3	25
W	45	7	67	98	32	54

Veja o resultado do mapa do processador, onde cada letra corresponde a 1 seg. Espaços são só para clareza e separam 10 segundos.

```
 HHUUUUUU UU      MM MMMMIIIQQQ MOOORRRIIE EEQQQ
BBBYY YJJJRRRPPP SSSNNNWWWB YYYJJJRRPP PSSSWWWYYY
 JPPPSSSWYP PSS
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 22.9 segundos e 33.9 segundos respectivamente.

☞ Para você fazer

Imagine a seguinte situação em um ambiente multiprogramado com o método **Round Robin, com q=3** .

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
K	5	4				
R	6	5				
Z	14	11				
W	18	3				
M	19	3				
A	22	9				
T	26	7				
N	27	9				
H	30	8				
O	34	11				
J	35	8				
P	39	11				
L	42	8				
V	43	2				
Q	45	8				

Para preencher a tabela acima, use o seguinte espaço de tempo

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
100	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150

Responda aqui, qual a média de duração e de elapsed do exercício proposto com o método **Round Robin, com q=3** . (1 casa decimal, por favor)

duração	elapsed

CEP - UP - UTFPR - PUCPr - UFPR –
03/06/2024 - 13:31:33.3
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V:
2.02 76575 LAURA PACHECO DELATTRE
24CC2204 - 9 _____ / _____ / _____

Executando programas

Um programa é uma unidade de trabalho sendo executado em um computador. Em geral, um programa obtém dados de entrada, processa-os e gera dados de saída. Um detalhe importante é que (graças à arquitetura de Von Neumann) programas são tratados e armazenados na memória como se fossem dados.

A CPU utiliza ciclos de máquina repetitivos para executar as instruções do programa, uma a uma, do início ao fim. Cada ciclo é composto pelas fases de busca, decodificação e execução. Na fase de busca, a unidade de controle manda o sistema copiar no registrador de instruções a próxima instrução. (isto é feito a partir do registrador chamado contador do programa). Ao acabar esta fase o contador é incrementado para apontar para a próxima instrução. A fase de decodificação visa preparar os circuitos necessários para obedecer à instrução, seja ela qual for. Finalmente, a execução, é a implementação da instrução.

Note que as velocidades da CPU e dos dispositivos de E/S são completamente díspares, o que implica em algum tipo de sincronização (e obviamente abre as portas para inúmeros níveis de multiprocessamento). Três métodos são usados para obter sincronização: E/S programada (o mais primitivo: a CPU espera pelo dispositivo de E/S), E/S dirigido por interrupção, onde a CPU comanda a transferência, mas não fica aguardando sua conclusão. Aqui o dispositivo de E/S interrompe a CPU quando acabar. Durante este tempo, a CPU pode realizar outras tarefas. Finalmente, o método de Acesso Direto à memória (DMA), transfere um grande bloco de dados entre o dispositivo (rápido) de E/S e a memória, sem precisar passar pela CPU. Isto requer um controlador de DMA.

CPUS modernas
A seguir algumas descrições de processadores modernos, no estado da arte em 2024:

- CPU Intel Core i9-12900K: 5,5 GHz, 16 núcleos, 24 threads, até 5,5 GHz de clock boost, taxa de ciclo de máquina estimada entre 220 e 275 ciclos por instrução (CPI).
- CPU AMD Ryzen 9 5950X: 4,9 GHz, 16 núcleos, 32 threads, até 5,0 GHz de clock boost, taxa de ciclo de máquina estimada entre 200 e 250 CPI.
- CPU Apple M1 Pro: 3,2 GHz, 10 núcleos (8 núcleos de alto desempenho + 2 núcleos de alta eficiência), 16 threads, até 3,2 GHz de clock boost, taxa de ciclo de máquina estimada entre 150 e 200 CPI.

A taxa de ciclo é uma medida da eficiência ao executar uma instrução: ela indica o número de ciclos de relógio necessários para executar 1 instrução. O clock boost é uma execução acelerada que pode ser invocada pela CPU (depende de temperatura, carga, eficiência energética, etc)

Um núcleo é capaz de executar uma instrução por vez. Cada núcleo possui seus próprios recursos dedicados, como registradores e unidades de execução, permitindo que ele processe tarefas de forma independente. Quanto mais núcleos uma CPU possui ela pode lidar com mais tarefas simultaneamente.

As threads, por outro lado, dividem o tempo de processamento com os núcleos. Elas compartilham os mesmos recursos do núcleo, mas podem alternar a execução de diferentes instruções. Isso permite que um único núcleo execute várias threads ao mesmo tempo, aumentando ainda mais a capacidade multitarefa da CPU.

Uma analogia: Imagine uma cozinha como a CPU e os chefs como os núcleos. Cada chef (núcleo) tem sua própria estação de trabalho (recursos dedicados) e pode preparar um prato (instrução) por vez. As garçonetes (threads) circulam pela cozinha, pegando os pratos prontos dos chefs e entregando-os aos clientes (aplicativos). Quanto mais chefs (núcleos) e garçonetes (threads) a cozinha tiver, mais pratos (tarefas) ela poderá preparar e servir (executar) ao mesmo tempo.

Diferentes arquiteturas

CISC acrônimo de *Complex Instruction Set Computer*. Tem um grande conjunto de instruções, incluindo muitas complexas. Torna o processo de programação mais fácil, já que para muitas tarefas, há uma instrução nativa. A complexidade das instruções torna os circuitos da CPU e da UC mais complicados. A maneira de lidar com isso foi construir a programação em dois níveis. No primeiro, chamado micro-operações, estão apenas as operações muito simples, as únicas efetuadas diretamente pela CPU. Surge uma micromemória, que contém a tradução das micro-operações da operação complexa que está sendo executada. Como tudo na vida, há vantagens (programas menores no nível da máquina) e desvantagens (sobrecarga da micromemória e da micro-programação). Um exemplo desta arquitetura é a série Pentium da Intel, bem como mainframes e supercomputadores. Note-se que os PCs parecem estar migrando para a arquitetura RISC.

RISC Acrônimo de *Reduced Instruction Set Computer*. A idéia é ter o mínimo possível no conjunto de instruções de máquina, traduzindo operações complexas em macros usando operações simples sob controle do programa (e não da máquina como no caso CISC). Exemplos de computadores RISC: smartphones e tablets (processadores ARM), consoles de videogame (Nintendo Switch), Apple MacBook Air, entre outros.
Para encerrar vale dizer que mais recentemente computadores começaram a usar recursos de ambas as abordagens.

Pipeline Como se viu, um computador executa operações em 3 fases: busca, decodificação e execução. Originalmente, as 3 fases ocorriam em série para cada instrução: a instrução x precisava concluir as 3 fases antes de começar o ciclo de instrução $x+1$. Mais modernamente os computadores começaram a usar a técnica de *pipeline*, permitindo que a UC possa começar as fases da $x+1$ enquanto ainda trata a x.

Processamento paralelo Quando o custo do hardware começou a cair, computadores começaram a ser desenhados com várias UCs, várias ULAs e várias memórias. Isto permite o paralelismo na execução de instruções. Agora surgem 4 possibilidades: SISD (fluxo único de instruções e único de dados), SIMD (fluxo único de instruções e múltiplo de dados), MISD (único de dados e múltiplo de instruções) e MIMD (múltiplos ambos). Vale notar que o MISD vale como conceito, mas parece não ter sido nunca implementado. A idéia do processamento paralelo, parece mais fácil como conceito do que na prática. Há aqui um problema de balanceamento de carga: a alocação dinâmica de tarefas para que estas sejam executadas nos diversos processardores disponíveis, mas de forma que todos os processadores sejam utilizados adequadamente. Associado a isto está o problema do escalonamento, em inglês *scaling* (divisão de uma tarefa em sub-tarefas de acordo com o número de processadores disponíveis).

Com o aumento do número de tarefas, cresce exponencialmente o trabalho necessário para coordenar a interação entre elas. Se houver 4 tarefas, há potencialmente 6 pares de tarefas se comunicando. Em 5 tarefas, há 10 pares e em 6 tarefas, 15 pares.

Filas dentro do Sist. Operacional

Imagine-se um sistema operacional multiprogramado despachando tarefas. Como o processador é um só, e existem muitos processos querendo-o usar, formam-se filas. Alem do algoritmo FIFO também conhecido como FCFS (*fist come, first served*), vão ser usados 2 algoritmos distintos:

SJF *Shortest Job First.* Neste esquema, quando o processador vai atender alguém, ele escolhe na fila de espera, aquele que tiver a menor requisição de tempo. Se houver empate, o primeiro será retirado.

RR *Round Robin.* Neste esquema preemptivo, a fila será atendida em ordem, mas com uma quantidade fixa de tempo para cada processo. Após receber este *quantum*, o processo vai para o fim da fila.

Para este problema, haverá um padrão de chegadas de requisições de tempo. A unidade a ser considerada é o segundo. Não haverá nenhum tempo de *overhead* para troca de processos. Irão ser usados para cada processo, dois tempos, assim definidos:

duração do processo tempo de fim - tempo de início + 1

elapsed time tempo de fim - tempo de chegada + 1

Acompanhe no exemplo feito, usando o método **Round Robin, com q=3**

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
Q	1	8	1	11	11	11
W	5	10	7	50	44	46
A	7	6	12	23	12	17
F	8	9	15	43	29	36
O	16	2	24	25	2	10
I	20	9	29	72	44	53
G	22	3	35	37	3	16
C	23	8	38	84	47	62
K	33	8	47	88	42	56
X	35	4	51	76	26	42
Y	37	8	54	90	37	54
H	38	6	57	82	26	45
S	39	2	60	61	2	23
R	42	5	65	86	22	45
V	44	2	68	69	2	26

Veja o resultado do mapa do processador, onde cada letra corresponde a 1 seg. Espaços são só para clareza e separam 10 segundos.

```
QQQQQQWWWQ QAAAFFFWWW AAAOOFFFII IWWWGGGCCC FFFII
IKKKW XXXYYYHHHS SCCCRRRVVI IIKKKXYYYH HHCCRRKKYY
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 23.3 segundos e 36.1 segundos respectivamente.

☞ Para você fazer

Imagine a seguinte situação em um ambiente multiprogramado com o método **Round Robin, com q=3** .

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
G	9	4				
V	10	4				
S	13	2				
D	17	3				
Q	18	11				
J	23	9				
I	24	7				
P	25	9				
T	30	2				
Y	31	2				
F	33	7				
U	36	5				
C	39	11				
X	44	10				
B	45	4				

Para preencher a tabela acima, use o seguinte espaço de tempo

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
100	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150

Responda aqui, qual a média de duração e de elapsed do exercício proposto com o método **Round Robin, com q=3** . (1 casa decimal, por favor)

duração	elapsed

Executando programas

Um programa é uma unidade de trabalho sendo executado em um computador. Em geral, um programa obtém dados de entrada, processa-os e gera dados de saída. Um detalhe importante é que (graças à arquitetura de Von Neumann) programas são tratados e armazenados na memória como se fossem dados.

A CPU utiliza ciclos de máquina repetitivos para executar as instruções do programa, uma a uma, do início ao fim. Cada ciclo é composto pelas fases de busca, decodificação e execução. Na fase de busca, a unidade de controle manda o sistema copiar no registrador de instruções a próxima instrução. (isto é feito a partir do registrador chamado contador do programa). Ao acabar esta fase o contador é incrementado para apontar para a próxima instrução. A fase de decodificação visa preparar os circuitos necessários para obedecer à instrução, seja ela qual for. Finalmente, a execução, é a implementação da instrução.

Note que as velocidades da CPU e dos dispositivos de E/S são completamente díspares, o que implica em algum tipo de sincronização (e obviamente abre as portas para inúmeros níveis de multiprocessamento). Três métodos são usados para obter sincronização: E/S programada (o mais primitivo: a CPU espera pelo dispositivo de E/S), E/S dirigido por interrupção, onde a CPU comanda a transferência, mas não fica aguardando sua conclusão. Aqui o dispositivo de E/S interrompe a CPU quando acabar. Durante este tempo, a CPU pode realizar outras tarefas. Finalmente, o método de Acesso Direto à memória (DMA), transfere um grande bloco de dados entre o dispositivo (rápido) de E/S e a memória, sem precisar passar pela CPU. Isto requer um controlador de DMA.

CPUS modernas
A seguir algumas descrições de processadores modernos, no estado da arte em 2024:

- CPU Intel Core i9-12900K: 5,5 GHz, 16 núcleos, 24 threads, até 5,5 GHz de clock boost, taxa de ciclo de máquina estimada entre 220 e 275 ciclos por instrução (CPI).
- CPU AMD Ryzen 9 5950X: 4,9 GHz, 16 núcleos, 32 threads, até 5,0 GHz de clock boost, taxa de ciclo de máquina estimada entre 200 e 250 CPI.
- CPU Apple M1 Pro: 3,2 GHz, 10 núcleos (8 núcleos de alto desempenho + 2 núcleos de alta eficiência), 16 threads, até 3,2 GHz de clock boost, taxa de ciclo de máquina estimada entre 150 e 200 CPI.

A taxa de ciclo é uma medida da eficiência ao executar uma instrução: ela indica o número de ciclos de relógio necessários para executar 1 instrução. O clock boost é uma execução acelerada que pode ser invocada pela CPU (depende de temperatura, carga, eficiência energética, etc)

Um núcleo é capaz de executar uma instrução por vez. Cada núcleo possui seus próprios recursos dedicados, como registradores e unidades de execução, permitindo que ele processe tarefas de forma independente. Quanto mais núcleos uma CPU possui ela pode lidar com mais tarefas simultaneamente.

As threads, por outro lado, dividem o tempo de processamento com os núcleos. Elas compartilham os mesmos recursos do núcleo, mas podem alternar a execução de diferentes instruções. Isso permite que um único núcleo execute várias threads ao mesmo tempo, aumentando ainda mais a capacidade multitarefa da CPU.

Uma analogia: Imagine uma cozinha como a CPU e os chefs como os núcleos. Cada chef (núcleo) tem sua própria estação de trabalho (recursos dedicados) e pode preparar um prato (instrução) por vez. As garçonetes (threads) circulam pela cozinha, pegando os pratos prontos dos chefs e entregando-os aos clientes (aplicativos). Quanto mais chefs (núcleos) e garçonetes (threads) a cozinha tiver, mais pratos (tarefas) ela poderá preparar e servir (executar) ao mesmo tempo.

Diferentes arquiteturas

CISC acrônimo de *Complex Instruction Set Computer*. Tem um grande conjunto de instruções, incluindo muitas complexas. Torna o processo de programação mais fácil, já que para muitas tarefas, há uma instrução nativa. A complexidade das instruções torna os circuitos da CPU e da UC mais complicados. A maneira de lidar com isso foi construir a programação em dois níveis. No primeiro, chamado micro-operações, estão apenas as operações muito simples, as únicas efetuadas diretamente pela CPU. Surge uma micromemória, que contém a tradução das micro-operações da operação complexa que está sendo executada. Como tudo na vida, há vantagens (programas menores no nível da máquina) e desvantagens (sobrecarga da micromemória e da micro-programação). Um exemplo desta arquitetura é a série Pentium da Intel, bem como mainframes e supercomputadores. Note-se que os PCs parecem estar migrando para a arquitetura RISC.

RISC Acrônimo de *Reduced Instruction Set Computer*. A idéia é ter o mínimo possível no conjunto de instruções de máquina, traduzindo operações complexas em macros usando operações simples sob controle do programa (e não da máquina como no caso CISC). Exemplos de computadores RISC: smartphones e tablets (processadores ARM), consoles de videogame (Nintendo Switch), Apple MacBook Air, entre outros.
Para encerrar vale dizer que mais recentemente computadores começaram a usar recursos de ambas as abordagens.

Pipeline Como se viu, um computador executa operações em 3 fases: busca, decodificação e execução. Originalmente, as 3 fases ocorriam em série para cada instrução: a instrução x precisava concluir as 3 fases antes de começar o ciclo de instrução $x+1$. Mais modernamente os computadores começaram a usar a técnica de *pipeline*, permitindo que a UC possa começar as fases da $x+1$ enquanto ainda trata a x.

Processamento paralelo Quando o custo do hardware começou a cair, computadores começaram a ser desenhados com várias UCs, várias ULAs e várias memórias. Isto permite o paralelismo na execução de instruções. Agora surgem 4 possibilidades: SISD (fluxo único de instruções e único de dados), SIMD (fluxo único de instruções e múltiplo de dados), MISD (único de dados e múltiplo de instruções) e MIMD (múltiplos ambos). Vale notar que o MISD vale como conceito, mas parece não ter sido nunca implementado. A idéia do processamento paralelo, parece mais fácil como conceito do que na prática. Há aqui um problema de balanceamento de carga: a alocação dinâmica de tarefas para que estas sejam executadas nos diversos processardores disponíveis, mas de forma que todos os processadores sejam utilizados adequadamente. Associado a isto está o problema do escalonamento, em inglês *scaling* (divisão de uma tarefa em sub-tarefas de acordo com o número de processadores disponíveis).

Com o aumento do número de tarefas, cresce exponencialmente o trabalho necessário para coordenar a interação entre elas. Se houver 4 tarefas, há potencialmente 6 pares de tarefas se comunicando. Em 5 tarefas, há 10 pares e em 6 tarefas, 15 pares.

Filas dentro do Sist. Operacional

Imagine-se um sistema operacional multiprogramado despachando tarefas. Como o processador é um só, e existem muitos processos querendo-o usar, formam-se filas. Alem do algoritmo FIFO também conhecido como FCFS (*fist come, first served*), vão ser usados 2 algoritmos distintos:

SJF *Shortest Job First.* Neste esquema, quando o processador vai atender alguém, ele escolhe na fila de espera, aquele que tiver a menor requisição de tempo. Se houver empate, o primeiro será retirado.

RR *Round Robin.* Neste esquema preemptivo, a fila será atendida em ordem, mas com uma quantidade fixa de tempo para cada processo. Após receber este *quantum*, o processo vai para o fim da fila.

Para este problema, haverá um padrão de chegadas de requisições de tempo. A unidade a ser considerada é o segundo. Não haverá nenhum tempo de *overhead* para troca de processos. Irão ser usados para cada processo, dois tempos, assim definidos:

duração do processo tempo de fim - tempo de início + 1

elapsed time tempo de fim - tempo de chegada + 1

Acompanhe no exemplo feito, usando o método **Round Robin, com q=3**

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
H	1	4	1	7	7	7
M	3	10	4	26	23	24
I	8	10	11	77	67	70
A	11	9	17	72	56	62
C	13	5	20	40	21	28
G	17	7	27	84	58	68
R	18	11	30	102	73	85
U	19	9	33	93	61	75
D	25	4	41	76	36	52
W	27	9	47	99	53	73
X	29	7	50	100	51	72
L	33	2	59	60	2	28
E	35	8	61	104	44	70
O	36	3	67	69	3	34
S	41	7	73	105	33	65

Veja o resultado do mapa do processador, onde cada letra corresponde a 1 seg. Espaços são só para clareza e separam 10 segundos.

```
HHHMMMHMMM IIIMMMAAAC CCIIIMGGGR RRUUUAAACC DDDII
IWWWX XXGGGRRRLL EEEUUUOOOA AASSSDIWWW XXXGRRREEE
 UUUSSSWWWX RREES
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 39.2 segundos e 54.2 segundos respectivamente.

☞ Para você fazer

Imagine a seguinte situação em um ambiente multiprogramado com o método **Round Robin, com q=3** .

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
Z	2	9				
X	3	7				
R	8	11				
P	11	9				
F	14	9				
I	15	11				
H	18	4				
K	19	10				
Q	26	5				
V	33	7				
E	37	2				
W	40	11				
B	41	5				
O	43	11				
G	45	8				

Para preencher a tabela acima, use o seguinte espaço de tempo

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
100	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150

Responda aqui, qual a média de duração e de elapsed do exercício proposto com o método **Round Robin, com q=3** . (1 casa decimal, por favor)

duração	elapsed

CEP - UP - UTFPR - PUCPr - UFPR –
03/06/2024 - 13:31:33.3
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V:
2.02 76599 NICOLE DA SILVA DE AZEVEDO
24CC2204 - 11 _____ / _____ / _____

Executando programas

Um programa é uma unidade de trabalho sendo executado em um computador. Em geral, um programa obtém dados de entrada, processa-os e gera dados de saída. Um detalhe importante é que (graças à arquitetura de Von Neumann) programas são tratados e armazenados na memória como se fossem dados.

A CPU utiliza ciclos de máquina repetitivos para executar as instruções do programa, uma a uma, do início ao fim. Cada ciclo é composto pelas fases de busca, decodificação e execução. Na fase de busca, a unidade de controle manda o sistema copiar no registrador de instruções a próxima instrução. (isto é feito a partir do registrador chamado contador do programa). Ao acabar esta fase o contador é incrementado para apontar para a próxima instrução. A fase de decodificação visa preparar os circuitos necessários para obedecer à instrução, seja ela qual for. Finalmente, a execução, é a implementação da instrução.

Note que as velocidades da CPU e dos dispositivos de E/S são completamente díspares, o que implica em algum tipo de sincronização (e obviamente abre as portas para inúmeros níveis de multiprocessamento). Três métodos são usados para obter sincronização: E/S programada (o mais primitivo: a CPU espera pelo dispositivo de E/S), E/S dirigido por interrupção, onde a CPU comanda a transferência, mas não fica aguardando sua conclusão. Aqui o dispositivo de E/S interrompe a CPU quando acabar. Durante este tempo, a CPU pode realizar outras tarefas. Finalmente, o método de Acesso Direto à memória (DMA), transfere um grande bloco de dados entre o dispositivo (rápido) de E/S e a memória, sem precisar passar pela CPU. Isto requer um controlador de DMA.

CPUS modernas
A seguir algumas descrições de processadores modernos, no estado da arte em 2024:

- CPU Intel Core i9-12900K: 5,5 GHz, 16 núcleos, 24 threads, até 5,5 GHz de clock boost, taxa de ciclo de máquina estimada entre 220 e 275 ciclos por instrução (CPI).
- CPU AMD Ryzen 9 5950X: 4,9 GHz, 16 núcleos, 32 threads, até 5,0 GHz de clock boost, taxa de ciclo de máquina estimada entre 200 e 250 CPI.
- CPU Apple M1 Pro: 3,2 GHz, 10 núcleos (8 núcleos de alto desempenho + 2 núcleos de alta eficiência), 16 threads, até 3,2 GHz de clock boost, taxa de ciclo de máquina estimada entre 150 e 200 CPI.

A taxa de ciclo é uma medida da eficiência ao executar uma instrução: ela indica o número de ciclos de relógio necessários para executar 1 instrução. O clock boost é uma execução acelerada que pode ser invocada pela CPU (depende de temperatura, carga, eficiência energética, etc)

Um núcleo é capaz de executar uma instrução por vez. Cada núcleo possui seus próprios recursos dedicados, como registradores e unidades de execução, permitindo que ele processe tarefas de forma independente. Quanto mais núcleos uma CPU possui ela pode lidar com mais tarefas simultaneamente.

As threads, por outro lado, dividem o tempo de processamento com os núcleos. Elas compartilham os mesmos recursos do núcleo, mas podem alternar a execução de diferentes instruções. Isso permite que um único núcleo execute várias threads ao mesmo tempo, aumentando ainda mais a capacidade multitarefa da CPU.

Uma analogia: Imagine uma cozinha como a CPU e os chefs como os núcleos. Cada chef (núcleo) tem sua própria estação de trabalho (recursos dedicados) e pode preparar um prato (instrução) por vez. As garçonetes (threads) circulam pela cozinha, pegando os pratos prontos dos chefs e entregando-os aos clientes (aplicativos). Quanto mais chefs (núcleos) e garçonetes (threads) a cozinha tiver, mais pratos (tarefas) ela poderá preparar e servir (executar) ao mesmo tempo.

Diferentes arquiteturas

CISC acrônimo de *Complex Instruction Set Computer*. Tem um grande conjunto de instruções, incluindo muitas complexas. Torna o processo de programação mais fácil, já que para muitas tarefas, há uma instrução nativa. A complexidade das instruções torna os circuitos da CPU e da UC mais complicados. A maneira de lidar com isso foi construir a programação em dois níveis. No primeiro, chamado micro-operações, estão apenas as operações muito simples, as únicas efetuadas diretamente pela CPU. Surge uma micromemória, que contém a tradução das micro-operações da operação complexa que está sendo executada. Como tudo na vida, há vantagens (programas menores no nível da máquina) e desvantagens (sobrecarga da micromemória e da micro-programação). Um exemplo desta arquitetura é a série Pentium da Intel, bem como mainframes e supercomputadores. Note-se que os PCs parecem estar migrando para a arquitetura RISC.

RISC Acrônimo de *Reduced Instruction Set Computer*. A idéia é ter o mínimo possível no conjunto de instruções de máquina, traduzindo operações complexas em macros usando operações simples sob controle do programa (e não da máquina como no caso CISC). Exemplos de computadores RISC: smartphones e tablets (processadores ARM), consoles de videogame (Nintendo Switch), Apple MacBook Air, entre outros.
Para encerrar vale dizer que mais recentemente computadores começaram a usar recursos de ambas as abordagens.

Pipeline Como se viu, um computador executa operações em 3 fases: busca, decodificação e execução. Originalmente, as 3 fases ocorriam em série para cada instrução: a instrução x precisava concluir as 3 fases antes de começar o ciclo de instrução $x+1$. Mais modernamente os computadores começaram a usar a técnica de *pipeline*, permitindo que a UC possa começar as fases da $x+1$ enquanto ainda trata a x.

Processamento paralelo Quando o custo do hardware começou a cair, computadores começaram a ser desenhados com várias UCs, várias ULAs e várias memórias. Isto permite o paralelismo na execução de instruções. Agora surgem 4 possibilidades: SISD (fluxo único de instruções e único de dados), SIMD (fluxo único de instruções e múltiplo de dados), MISD (único de dados e múltiplo de instruções) e MIMD (múltiplos ambos). Vale notar que o MISD vale como conceito, mas parece não ter sido nunca implementado. A idéia do processamento paralelo, parece mais fácil como conceito do que na prática. Há aqui um problema de balanceamento de carga: a alocação dinâmica de tarefas para que estas sejam executadas nos diversos processardores disponíveis, mas de forma que todos os processadores sejam utilizados adequadamente. Associado a isto está o problema do escalonamento, em inglês *scaling* (divisão de uma tarefa em sub-tarefas de acordo com o número de processadores disponíveis).

Com o aumento do número de tarefas, cresce exponencialmente o trabalho necessário para coordenar a interação entre elas. Se houver 4 tarefas, há potencialmente 6 pares de tarefas se comunicando. Em 5 tarefas, há 10 pares e em 6 tarefas, 15 pares.

Filas dentro do Sist. Operacional

Imagine-se um sistema operacional multiprogramado despachando tarefas. Como o processador é um só, e existem muitos processos querendo-o usar, formam-se filas. Alem do algoritmo FIFO também conhecido como FCFS (*fist come, first served*), vão ser usados 2 algoritmos distintos:

SJF *Shortest Job First*. Neste esquema, quando o processador vai atender alguém, ele escolhe na fila de espera, aquele que tiver a menor requisição de tempo. Se houver empate, o primeiro será retirado.

RR *Round Robin*. Neste esquema preemptivo, a fila será atendida em ordem, mas com uma quantidade fixa de tempo para cada processo. Após receber este *quantum*, o processo vai para o fim da fila.

Para este problema, haverá um padrão de chegadas de requisições de tempo. A unidade a ser considerada é o segundo. Não haverá nenhum tempo de *overhead* para troca de processos. Irão ser usados para cada processo, dois tempos, assim definidos:

duração do processo tempo de fim - tempo de início + 1

elapsed time tempo de fim - tempo de chegada + 1

Acompanhe no exemplo feito, usando o método **Shortest Job First**

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
X	2	9	2	10	9	9
Y	3	3	11	13	3	11
F	4	11	81	91	11	88
I	7	5	14	18	5	12
M	8	6	23	28	6	21
V	9	10	61	70	10	62
N	18	4	19	22	4	5
E	19	11	92	102	11	84
G	20	7	45	51	7	32
D	27	10	71	80	10	54
K	28	4	31	34	4	7
S	29	2	29	30	2	2
L	33	6	35	40	6	8
O	37	4	41	44	4	8
Q	41	9	52	60	9	20

Veja o resultado do mapa do processador, onde cada letra corresponde a 1 seg. Espaços são só para clareza e separam 10 segundos.

```
 XXXXXXXXX YYYIIIIIINN NNMMMMMMSS KKKKLLLLLL OOOOG
GGGGG GQQQQQQQQQ VVVVVVVVVV DDDDDDDDD FFFFFFFFF
 FEEEEEEEE EE
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 6.7 segundos e 28.2 segundos respectivamente.

☞ Para você fazer

Imagine a seguinte situação em um ambiente multiprogramado com o método **Shortest Job First** .

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
Y	1	10				
A	5	4				
F	8	6				
J	13	10				
E	15	9				
U	18	9				
X	20	6				
Z	22	6				
L	24	5				
B	28	11				
H	34	2				
T	35	8				
V	39	4				
O	41	8				
K	45	6				

Para preencher a tabela acima, use o seguinte espaço de tempo

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
100	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150

Responda aqui, qual a média de duração e de elapsed do exercício proposto com o método **Shortest Job First** . (1 casa decimal, por favor)

duração	elapsed

CEP - UP - UTFPR - PUCPr - UFPR –
03/06/2024 - 13:31:33.3
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V:
2.02 76601 PAULO JOSE FREIRE CORREA
24CC2204 - 12 _____ / _____ / _____

Executando programas

Um programa é uma unidade de trabalho sendo executado em um computador. Em geral, um programa obtém dados de entrada, processa-os e gera dados de saída. Um detalhe importante é que (graças à arquitetura de Von Neumann) programas são tratados e armazenados na memória como se fossem dados.

A CPU utiliza ciclos de máquina repetitivos para executar as instruções do programa, uma a uma, do início ao fim. Cada ciclo é composto pelas fases de busca, decodificação e execução. Na fase de busca, a unidade de controle manda o sistema copiar no registrador de instruções a próxima instrução. (isto é feito a partir do registrador chamado contador do programa). Ao acabar esta fase o contador é incrementado para apontar para a próxima instrução. A fase de decodificação visa preparar os circuitos necessários para obedecer à instrução, seja ela qual for. Finalmente, a execução, é a implementação da instrução.

Note que as velocidades da CPU e dos dispositivos de E/S são completamente díspares, o que implica em algum tipo de sincronização (e obviamente abre as portas para inúmeros níveis de multiprocessamento). Três métodos são usados para obter sincronização: E/S programada (o mais primitivo: a CPU espera pelo dispositivo de E/S), E/S dirigido por interrupção, onde a CPU comanda a transferência, mas não fica aguardando sua conclusão. Aqui o dispositivo de E/S interrompe a CPU quando acabar. Durante este tempo, a CPU pode realizar outras tarefas. Finalmente, o método de Acesso Direto à memória (DMA), transfere um grande bloco de dados entre o dispositivo (rápido) de E/S e a memória, sem precisar passar pela CPU. Isto requer um controlador de DMA.

CPUS modernas
A seguir algumas descrições de processadores modernos, no estado da arte em 2024:

- CPU Intel Core i9-12900K: 5,5 GHz, 16 núcleos, 24 threads, até 5,5 GHz de clock boost, taxa de ciclo de máquina estimada entre 220 e 275 ciclos por instrução (CPI).
- CPU AMD Ryzen 9 5950X: 4,9 GHz, 16 núcleos, 32 threads, até 5,0 GHz de clock boost, taxa de ciclo de máquina estimada entre 200 e 250 CPI.
- CPU Apple M1 Pro: 3,2 GHz, 10 núcleos (8 núcleos de alto desempenho + 2 núcleos de alta eficiência), 16 threads, até 3,2 GHz de clock boost, taxa de ciclo de máquina estimada entre 150 e 200 CPI.

A taxa de ciclo é uma medida da eficiência ao executar uma instrução: ela indica o número de ciclos de relógio necessários para executar 1 instrução. O clock boost é uma execução acelerada que pode ser invocada pela CPU (depende de temperatura, carga, eficiência energética, etc)

Um núcleo é capaz de executar uma instrução por vez. Cada núcleo possui seus próprios recursos dedicados, como registradores e unidades de execução, permitindo que ele processe tarefas de forma independente. Quanto mais núcleos uma CPU possui ela pode lidar com mais tarefas simultaneamente.

As threads, por outro lado, dividem o tempo de processamento com os núcleos. Elas compartilham os mesmos recursos do núcleo, mas podem alternar a execução de diferentes instruções. Isso permite que um único núcleo execute várias threads ao mesmo tempo, aumentando ainda mais a capacidade multitarefa da CPU.

Uma analogia: Imagine uma cozinha como a CPU e os chefs como os núcleos. Cada chef (núcleo) tem sua própria estação de trabalho (recursos dedicados) e pode preparar um prato (instrução) por vez. As garçonetes (threads) circulam pela cozinha, pegando os pratos prontos dos chefs e entregando-os aos clientes (aplicativos). Quanto mais chefs (núcleos) e garçonetes (threads) a cozinha tiver, mais pratos (tarefas) ela poderá preparar e servir (executar) ao mesmo tempo.

Diferentes arquiteturas

CISC acrônimo de *Complex Instruction Set Computer*. Tem um grande conjunto de instruções, incluindo muitas complexas. Torna o processo de programação mais fácil, já que para muitas tarefas, há uma instrução nativa. A complexidade das instruções torna os circuitos da CPU e da UC mais complicados. A maneira de lidar com isso foi construir a programação em dois níveis. No primeiro, chamado micro-operações, estão apenas as operações muito simples, as únicas efetuadas diretamente pela CPU. Surge uma micromemória, que contém a tradução das micro-operações da operação complexa que está sendo executada. Como tudo na vida, há vantagens (programas menores no nível da máquina) e desvantagens (sobrecarga da micromemória e da micro-programação). Um exemplo desta arquitetura é a série Pentium da Intel, bem como mainframes e supercomputadores. Note-se que os PCs parecem estar migrando para a arquitetura RISC.

RISC Acrônimo de *Reduced Instruction Set Computer*. A idéia é ter o mínimo possível no conjunto de instruções de máquina, traduzindo operações complexas em macros usando operações simples sob controle do programa (e não da máquina como no caso CISC). Exemplos de computadores RISC: smartphones e tablets (processadores ARM), consoles de videogame (Nintendo Switch), Apple MacBook Air, entre outros.
Para encerrar vale dizer que mais recentemente computadores começaram a usar recursos de ambas as abordagens.

Pipeline Como se viu, um computador executa operações em 3 fases: busca, decodificação e execução. Originalmente, as 3 fases ocorriam em série para cada instrução: a instrução x precisava concluir as 3 fases antes de começar o ciclo de instrução $x+1$. Mais modernamente os computadores começaram a usar a técnica de *pipeline*, permitindo que a UC possa começar as fases da $x+1$ enquanto ainda trata a x.

Processamento paralelo Quando o custo do hardware começou a cair, computadores começaram a ser desenhados com várias UCs, várias ULAs e várias memórias. Isto permite o paralelismo na execução de instruções. Agora surgem 4 possibilidades: SISD (fluxo único de instruções e único de dados), SIMD (fluxo único de instruções e múltiplo de dados), MISD (único de dados e múltiplo de instruções) e MIMD (múltiplos ambos). Vale notar que o MISD vale como conceito, mas parece não ter sido nunca implementado. A idéia do processamento paralelo, parece mais fácil como conceito do que na prática. Há aqui um problema de balanceamento de carga: a alocação dinâmica de tarefas para que estas sejam executadas nos diversos processardores disponíveis, mas de forma que todos os processadores sejam utilizados adequadamente. Associado a isto está o problema do escalonamento, em inglês *scaling* (divisão de uma tarefa em sub-tarefas de acordo com o número de processadores disponíveis).

Com o aumento do número de tarefas, cresce exponencialmente o trabalho necessário para coordenar a interação entre elas. Se houver 4 tarefas, há potencialmente 6 pares de tarefas se comunicando. Em 5 tarefas, há 10 pares e em 6 tarefas, 15 pares.

Filas dentro do Sist. Operacional

Imagine-se um sistema operacional multiprogramado despachando tarefas. Como o processador é um só, e existem muitos processos querendo-o usar, formam-se filas. Alem do algoritmo FIFO também conhecido como FCFS (*fist come, first served*), vão ser usados 2 algoritmos distintos:

SJF *Shortest Job First.* Neste esquema, quando o processador vai atender alguém, ele escolhe na fila de espera, aquele que tiver a menor requisição de tempo. Se houver empate, o primeiro será retirado.

RR *Round Robin.* Neste esquema preemptivo, a fila será atendida em ordem, mas com uma quantidade fixa de tempo para cada processo. Após receber este *quantum*, o processo vai para o fim da fila.

Para este problema, haverá um padrão de chegadas de requisições de tempo. A unidade a ser considerada é o segundo. Não haverá nenhum tempo de *overhead* para troca de processos. Irão ser usados para cada processo, dois tempos, assim definidos:

duração do processo tempo de fim - tempo de início + 1

elapsed time tempo de fim - tempo de chegada + 1

Acompanhe no exemplo feito, usando o método **Shortest Job First**

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
G	2	10	2	11	10	10
N	3	9	53	61	9	59
B	4	9	62	70	9	67
O	5	3	12	14	3	10
Y	7	3	15	17	3	11
A	8	9	71	79	9	72
P	10	3	18	20	3	11
D	12	9	80	88	9	77
X	19	4	21	24	4	6
V	22	6	25	30	6	9
F	25	6	37	42	6	18
K	26	4	33	36	4	11
E	31	2	31	32	2	2
Z	38	2	43	44	2	7
C	42	8	45	52	8	11

Veja o resultado do mapa do processador, onde cada letra corresponde a 1 seg. Espaços são só para clareza e separam 10 segundos.

```
GGGGGGGGG GOOOYYYPPP XXXXVVVVVV EEKKKKFFFF FFZZC
CCCCC CCNNNNNNNN NBBBBBBBBB AAAAAAAAAD DDDDDDDD
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 5.8 segundos e 25.4 segundos respectivamente.

☞ Para você fazer

Imagine a seguinte situação em um ambiente multiprogramado com o método **Shortest Job First** .

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
G	3	3				
V	6	9				
B	7	8				
A	8	6				
U	13	3				
Q	15	11				
K	16	4				
O	18	4				
W	21	2				
R	22	8				
F	27	5				
Z	33	4				
T	34	7				
D	35	2				
P	36	2				

Para preencher a tabela acima, use o seguinte espaço de tempo

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
100	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150

Responda aqui, qual a média de duração e de elapsed do exercício proposto com o método **Shortest Job First** . (1 casa decimal, por favor)

duração	elapsed

CEP - UP - UTFPR - PUCPr - UFPR –
03/06/2024 - 13:31:33.3
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V: 2.02 76618 PEDRO HENRIQUE PIEKARSKI DE OL 24CC2204 - 13 _____ / _____ / _____

Executando programas

Um programa é uma unidade de trabalho sendo executado em um computador. Em geral, um programa obtém dados de entrada, processa-os e gera dados de saída. Um detalhe importante é que (graças à arquitetura de Von Neumann) programas são tratados e armazenados na memória como se fossem dados.

A CPU utiliza ciclos de máquina repetitivos para executar as instruções do programa, uma a uma, do início ao fim. Cada ciclo é composto pelas fases de busca, decodificação e execução. Na fase de busca, a unidade de controle manda o sistema copiar no registrador de instruções a próxima instrução. (isto é feito a partir do registrador chamado contador do programa). Ao acabar esta fase o contador é incrementado para apontar para a próxima instrução. A fase de decodificação visa preparar os circuitos necessários para obedecer à instrução, seja ela qual for. Finalmente, a execução, é a implementação da instrução.

Note que as velocidades da CPU e dos dispositivos de E/S são completamente díspares, o que implica em algum tipo de sincronização (e obviamente abre as portas para inúmeros níveis de multiprocessamento). Três métodos são usados para obter sincronização: E/S programada (o mais primitivo: a CPU espera pelo dispositivo de E/S), E/S dirigido por interrupção, onde a CPU comanda a transferência, mas não fica aguardando sua conclusão. Aqui o dispositivo de E/S interrompe a CPU quando acabar. Durante este tempo, a CPU pode realizar outras tarefas. Finalmente, o método de Acesso Direto à memória (DMA), transfere um grande bloco de dados entre o dispositivo (rápido) de E/S e a memória, sem precisar passar pela CPU. Isto requer um controlador de DMA.

CPUS modernas
A seguir algumas descrições de processadores modernos, no estado da arte em 2024:

- CPU Intel Core i9-12900K: 5,5 GHz, 16 núcleos, 24 threads, até 5,5 GHz de clock boost, taxa de ciclo de máquina estimada entre 220 e 275 ciclos por instrução (CPI).
- CPU AMD Ryzen 9 5950X: 4,9 GHz, 16 núcleos, 32 threads, até 5,0 GHz de clock boost, taxa de ciclo de máquina estimada entre 200 e 250 CPI.
- CPU Apple M1 Pro: 3,2 GHz, 10 núcleos (8 núcleos de alto desempenho + 2 núcleos de alta eficiência), 16 threads, até 3,2 GHz de clock boost, taxa de ciclo de máquina estimada entre 150 e 200 CPI.

A taxa de ciclo é uma medida da eficiência ao executar uma instrução: ela indica o número de ciclos de relógio necessários para executar 1 instrução. O clock boost é uma execução acelerada que pode ser invocada pela CPU (depende de temperatura, carga, eficiência energética, etc)

Um núcleo é capaz de executar uma instrução por vez. Cada núcleo possui seus próprios recursos dedicados, como registradores e unidades de execução, permitindo que ele processe tarefas de forma independente. Quanto mais núcleos uma CPU possui ela pode lidar com mais tarefas simultaneamente.

As threads, por outro lado, dividem o tempo de processamento com os núcleos. Elas compartilham os mesmos recursos do núcleo, mas podem alternar a execução de diferentes instruções. Isso permite que um único núcleo execute várias threads ao mesmo tempo, aumentando ainda mais a capacidade multitarefa da CPU.

Uma analogia: Imagine uma cozinha como a CPU e os chefs como os núcleos. Cada chef (núcleo) tem sua própria estação de trabalho (recursos dedicados) e pode preparar um prato (instrução) por vez. As garçonetes (threads) circulam pela cozinha, pegando os pratos prontos dos chefs e entregando-os aos clientes (aplicativos). Quanto mais chefs (núcleos) e garçonetes (threads) a cozinha tiver, mais pratos (tarefas) ela poderá preparar e servir (executar) ao mesmo tempo.

Diferentes arquiteturas

CISC acrônimo de *Complex Instruction Set Computer*. Tem um grande conjunto de instruções, incluindo muitas complexas. Torna o processo de programação mais fácil, já que para muitas tarefas, há uma instrução nativa. A complexidade das instruções torna os circuitos da CPU e da UC mais complicados. A maneira de lidar com isso foi construir a programação em dois níveis. No primeiro, chamado micro-operações, estão apenas as operações muito simples, as únicas efetuadas diretamente pela CPU. Surge uma micromemória, que contém a tradução das micro-operações da operação complexa que está sendo executada. Como tudo na vida, há vantagens (programas menores no nível da máquina) e desvantagens (sobrecarga da micromemória e da micro-programação). Um exemplo desta arquitetura é a série Pentium da Intel, bem como mainframes e supercomputadores. Note-se que os PCs parecem estar migrando para a arquitetura RISC.

RISC Acrônimo de *Reduced Instruction Set Computer*. A idéia é ter o mínimo possível no conjunto de instruções de máquina, traduzindo operações complexas em macros usando operações simples sob controle do programa (e não da máquina como no caso CISC). Exemplos de computadores RISC: smartphones e tablets (processadores ARM), consoles de videogame (Nintendo Switch), Apple MacBook Air, entre outros.
Para encerrar vale dizer que mais recentemente computadores começaram a usar recursos de ambas as abordagens.

Pipeline Como se viu, um computador executa operações em 3 fases: busca, decodificação e execução. Originalmente, as 3 fases ocorriam em série para cada instrução: a instrução x precisava concluir as 3 fases antes de começar o ciclo de instrução $x+1$. Mais modernamente os computadores começaram a usar a técnica de *pipeline*, permitindo que a UC possa começar as fases da $x+1$ enquanto ainda trata a x.

Processamento paralelo Quando o custo do hardware começou a cair, computadores começaram a ser desenhados com várias UCs, várias ULAs e várias memórias. Isto permite o paralelismo na execução de instruções. Agora surgem 4 possibilidades: SISD (fluxo único de instruções e único de dados), SIMD (fluxo único de instruções e múltiplo de dados), MISD (único de dados e múltiplo de instruções) e MIMD (múltiplos ambos). Vale notar que o MISD vale como conceito, mas parece não ter sido nunca implementado. A idéia do processamento paralelo, parece mais fácil como conceito do que na prática. Há aqui um problema de balanceamento de carga: a alocação dinâmica de tarefas para que estas sejam executadas nos diversos processardores disponíveis, mas de forma que todos os processadores sejam utilizados adequadamente. Associado a isto está o problema do escalonamento, em inglês *scaling* (divisão de uma tarefa em sub-tarefas de acordo com o número de processadores disponíveis).

Com o aumento do número de tarefas, cresce exponencialmente o trabalho necessário para coordenar a interação entre elas. Se houver 4 tarefas, há potencialmente 6 pares de tarefas se comunicando. Em 5 tarefas, há 10 pares e em 6 tarefas, 15 pares.

Filas dentro do Sist. Operacional

Imagine-se um sistema operacional multiprogramado despachando tarefas. Como o processador é um só, e existem muitos processos querendo-o usar, formam-se filas. Alem do algoritmo FIFO também conhecido como FCFS (*fist come, first served*), vão ser usados 2 algoritmos distintos:

SJF *Shortest Job First.* Neste esquema, quando o processador vai atender alguém, ele escolhe na fila de espera, aquele que tiver a menor requisição de tempo. Se houver empate, o primeiro será retirado.

RR *Round Robin.* Neste esquema preemptivo, a fila será atendida em ordem, mas com uma quantidade fixa de tempo para cada processo. Após receber este *quantum*, o processo vai para o fim da fila.

Para este problema, haverá um padrão de chegadas de requisições de tempo. A unidade a ser considerada é o segundo. Não haverá nenhum tempo de *overhead* para troca de processos. Irão ser usados para cada processo, dois tempos, assim definidos:

duração do processo tempo de fim - tempo de início + 1

elapsed time tempo de fim - tempo de chegada + 1

Acompanhe no exemplo feito, usando o método **Round Robin, com q=3**

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
T	9	10	9	40	32	32
K	11	4	12	21	10	11
F	14	7	18	58	41	45
X	18	11	25	108	84	91
U	19	5	28	54	27	36
R	21	11	34	110	77	90
J	23	6	37	67	31	45
S	26	7	41	101	61	76
Y	27	11	44	117	74	91
V	29	3	50	52	3	24
C	33	6	55	88	34	56
H	39	6	62	94	33	56
W	41	11	68	119	52	79
L	43	8	71	115	45	73
Z	45	5	77	103	27	59

Veja o resultado do mapa do processador, onde cada letra corresponde a 1 seg. Espaços são só para clareza e separam 10 segundos.

```
          TT TKKKTTTFFF KTTTXXXUUU FFFRRRJJJT SSSYY
YXXXV VVUUCCCFRR RHHHJJJWWW LLLSSSZZZY YYXXXCCCRR
 RHHHWWWLLL SZZYYYXXRR WWWLLYYWW
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 42.1 segundos e 57.6 segundos respectivamente.

☞ Para você fazer

Imagine a seguinte situação em um ambiente multiprogramado com o método **Round Robin, com q=3** .

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
O	7	11				
L	8	8				
B	12	9				
N	14	5				
Z	17	10				
C	19	4				
G	28	11				
S	29	4				
D	32	10				
R	33	4				
V	34	3				
Y	35	5				
M	38	3				
U	40	8				
F	45	5				

Para preencher a tabela acima, use o seguinte espaço de tempo

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
100	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150

Responda aqui, qual a média de duração e de elapsed do exercício proposto com o método **Round Robin, com q=3** . (1 casa decimal, por favor)

duração	elapsed

CEP - UP - UTFPR - PUCPr - UFPR – 03/06/2024 - 13:31:33.3
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V: 2.02 76625 RAUL KNAPKI CUNHA 24CC2204 - 14 _____ / _____ / _____

Executando programas

Um programa é uma unidade de trabalho sendo executado em um computador. Em geral, um programa obtém dados de entrada, processa-os e gera dados de saída. Um detalhe importante é que (graças à arquitetura de Von Neumann) programas são tratados e armazenados na memória como se fossem dados.

A CPU utiliza ciclos de máquina repetitivos para executar as instruções do programa, uma a uma, do início ao fim. Cada ciclo é composto pelas fases de busca, decodificação e execução. Na fase de busca, a unidade de controle manda o sistema copiar no registrador de instruções a próxima instrução. (isto é feito a partir do registrador chamado contador do programa). Ao acabar esta fase o contador é incrementado para apontar para a próxima instrução. A fase de decodificação visa preparar os circuitos necessários para obedecer à instrução, seja ela qual for. Finalmente, a execução, é a implementação da instrução.

Note que as velocidades da CPU e dos dispositivos de E/S são completamente díspares, o que implica em algum tipo de sincronização (e obviamente abre as portas para inúmeros níveis de multiprocessamento). Três métodos são usados para obter sincronização: E/S programada (o mais primitivo: a CPU espera pelo dispositivo de E/S), E/S dirigido por interrupção, onde a CPU comanda a transferência, mas não fica aguardando sua conclusão. Aqui o dispositivo de E/S interrompe a CPU quando acabar. Durante este tempo, a CPU pode realizar outras tarefas. Finalmente, o método de Acesso Direto à memória (DMA), transfere um grande bloco de dados entre o dispositivo (rápido) de E/S e a memória, sem precisar passar pela CPU. Isto requer um controlador de DMA.

CPUS modernas
A seguir algumas descrições de processadores modernos, no estado da arte em 2024:

- CPU Intel Core i9-12900K: 5,5 GHz, 16 núcleos, 24 threads, até 5,5 GHz de clock boost, taxa de ciclo de máquina estimada entre 220 e 275 ciclos por instrução (CPI).
- CPU AMD Ryzen 9 5950X: 4,9 GHz, 16 núcleos, 32 threads, até 5,0 GHz de clock boost, taxa de ciclo de máquina estimada entre 200 e 250 CPI.
- CPU Apple M1 Pro: 3,2 GHz, 10 núcleos (8 núcleos de alto desempenho + 2 núcleos de alta eficiência), 16 threads, até 3,2 GHz de clock boost, taxa de ciclo de máquina estimada entre 150 e 200 CPI.

A taxa de ciclo é uma medida da eficiência ao executar uma instrução: ela indica o número de ciclos de relógio necessários para executar 1 instrução. O clock boost é uma execução acelerada que pode ser invocada pela CPU (depende de temperatura, carga, eficiência energética, etc)

Um núcleo é capaz de executar uma instrução por vez. Cada núcleo possui seus próprios recursos dedicados, como registradores e unidades de execução, permitindo que ele processe tarefas de forma independente. Quanto mais núcleos uma CPU possui ela pode lidar com mais tarefas simultaneamente.

As threads, por outro lado, dividem o tempo de processamento com os núcleos. Elas compartilham os mesmos recursos do núcleo, mas podem alternar a execução de diferentes instruções. Isso permite que um único núcleo execute várias threads ao mesmo tempo, aumentando ainda mais a capacidade multitarefa da CPU.

Uma analogia: Imagine uma cozinha como a CPU e os chefs como os núcleos. Cada chef (núcleo) tem sua própria estação de trabalho (recursos dedicados) e pode preparar um prato (instrução) por vez. As garçonetes (threads) circulam pela cozinha, pegando os pratos prontos dos chefs e entregando-os aos clientes (aplicativos). Quanto mais chefs (núcleos) e garçonetes (threads) a cozinha tiver, mais pratos (tarefas) ela poderá preparar e servir (executar) ao mesmo tempo.

Diferentes arquiteturas

CISC acrônimo de *Complex Instruction Set Computer*. Tem um grande conjunto de instruções, incluindo muitas complexas. Torna o processo de programação mais fácil, já que para muitas tarefas, há uma instrução nativa. A complexidade das instruções torna os circuitos da CPU e da UC mais complicados. A maneira de lidar com isso foi construir a programação em dois níveis. No primeiro, chamado micro-operações, estão apenas as operações muito simples, as únicas efetuadas diretamente pela CPU. Surge uma micromemória, que contém a tradução das micro-operações da operação complexa que está sendo executada. Como tudo na vida, há vantagens (programas menores no nível da máquina) e desvantagens (sobrecarga da micromemória e da micro-programação). Um exemplo desta arquitetura é a série Pentium da Intel, bem como mainframes e supercomputadores. Note-se que os PCs parecem estar migrando para a arquitetura RISC.

RISC Acrônimo de *Reduced Instruction Set Computer*. A idéia é ter o mínimo possível no conjunto de instruções de máquina, traduzindo operações complexas em macros usando operações simples sob controle do programa (e não da máquina como no caso CISC). Exemplos de computadores RISC: smartphones e tablets (processadores ARM), consoles de videogame (Nintendo Switch), Apple MacBook Air, entre outros.
Para encerrar vale dizer que mais recentemente computadores começaram a usar recursos de ambas as abordagens.

Pipeline Como se viu, um computador executa operações em 3 fases: busca, decodificação e execução. Originalmente, as 3 fases ocorriam em série para cada instrução: a instrução x precisava concluir as 3 fases antes de começar o ciclo de instrução $x+1$. Mais modernamente os computadores começaram a usar a técnica de *pipeline*, permitindo que a UC possa começar as fases da $x+1$ enquanto ainda trata a x.

Processamento paralelo Quando o custo do hardware começou a cair, computadores começaram a ser desenhados com várias UCs, várias ULAs e várias memórias. Isto permite o paralelismo na execução de instruções. Agora surgem 4 possibilidades: SISD (fluxo único de instruções e único de dados), SIMD (fluxo único de instruções e múltiplo de dados), MISD (único de dados e múltiplo de instruções) e MIMD (múltiplos ambos). Vale notar que o MISD vale como conceito, mas parece não ter sido nunca implementado. A idéia do processamento paralelo, parece mais fácil como conceito do que na prática. Há aqui um problema de balanceamento de carga: a alocação dinâmica de tarefas para que estas sejam executadas nos diversos processardores disponíveis, mas de forma que todos os processadores sejam utilizados adequadamente. Associado a isto está o problema do escalonamento, em inglês *scaling* (divisão de uma tarefa em sub-tarefas de acordo com o número de processadores disponíveis).

Com o aumento do número de tarefas, cresce exponencialmente o trabalho necessário para coordenar a interação entre elas. Se houver 4 tarefas, há potencialmente 6 pares de tarefas se comunicando. Em 5 tarefas, há 10 pares e em 6 tarefas, 15 pares.

Filas dentro do Sist. Operacional

Imagine-se um sistema operacional multiprogramado despachando tarefas. Como o processador é um só, e existem muitos processos querendo-o usar, formam-se filas. Alem do algoritmo FIFO também conhecido como FCFS (*fist come, first served*), vão ser usados 2 algoritmos distintos:

SJF *Shortest Job First*. Neste esquema, quando o processador vai atender alguém, ele escolhe na fila de espera, aquele que tiver a menor requisição de tempo. Se houver empate, o primeiro será retirado.

RR *Round Robin*. Neste esquema preemptivo, a fila será atendida em ordem, mas com uma quantidade fixa de tempo para cada processo. Após receber este *quantum*, o processo vai para o fim da fila.

Para este problema, haverá um padrão de chegadas de requisições de tempo. A unidade a ser considerada é o segundo. Não haverá nenhum tempo de *overhead* para troca de processos. Irão ser usados para cada processo, dois tempos, assim definidos:

duração do processo tempo de fim - tempo de início + 1

elapsed time tempo de fim - tempo de chegada + 1

Acompanhe no exemplo feito, usando o método **Shortest Job First**

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
D	4	10	4	13	10	10
A	6	9	63	71	9	66
F	8	10	81	90	10	83
O	10	5	16	20	5	11
M	12	2	14	15	2	4
P	20	7	21	27	7	8
I	22	8	38	45	8	24
K	25	9	72	80	9	56
R	27	3	28	30	3	4
G	28	7	31	37	7	10
N	36	11	91	101	11	66
V	40	4	46	49	4	10
Z	41	7	56	62	7	22
T	43	11	102	112	11	70
U	44	6	50	55	6	12

Veja o resultado do mapa do processador, onde cada letra corresponde a 1 seg. Espaços são só para clareza e separam 10 segundos.

```
   DDDDDDD DDDMMOOOOO PPPPPPPRRR GGGGGGGIII IIIII
VVVVU UUUUUZZZZZ ZZAAAAAAAAA AKKKKKKKKK FFFFFFFFFF
 NNNNNNNNN NTTTTTTTTT TT
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 7.3 segundos e 30.4 segundos respectivamente.

☞ Para você fazer

Imagine a seguinte situação em um ambiente multiprogramado com o método **Shortest Job First** .

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
Z	3	3				
G	5	8				
J	6	10				
T	13	3				
B	19	2				
O	25	5				
Q	26	6				
R	28	10				
V	30	8				
C	31	9				
M	33	8				
S	39	4				
X	40	11				
E	42	8				
L	43	8				

Para preencher a tabela acima, use o seguinte espaço de tempo

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
100	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150

Responda aqui, qual a média de duração e de elapsed do exercício proposto com o método **Shortest Job First** . (1 casa decimal, por favor)

duração	elapsed

204-76625 - /

CEP - UP - UTFPR - PUCPr - UFPR –
03/06/2024 - 13:31:33.3
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V:
2.02 76632 VICTOR BRONDANI DOS SANTOS
24CC2204 - 15 _____ / _____ / _____

Executando programas

Um programa é uma unidade de trabalho sendo executado em um computador. Em geral, um programa obtém dados de entrada, processa-os e gera dados de saída. Um detalhe importante é que (graças à arquitetura de Von Neumann) programas são tratados e armazenados na memória como se fossem dados.

A CPU utiliza ciclos de máquina repetitivos para executar as instruções do programa, uma a uma, do início ao fim. Cada ciclo é composto pelas fases de busca, decodificação e execução. Na fase de busca, a unidade de controle manda o sistema copiar no registrador de instruções a próxima instrução. (isto é feito a partir do registrador chamado contador do programa). Ao acabar esta fase o contador é incrementado para apontar para a próxima instrução. A fase de decodificação visa preparar os circuitos necessários para obedecer à instrução, seja ela qual for. Finalmente, a execução, é a implementação da instrução.

Note que as velocidades da CPU e dos dispositivos de E/S são completamente díspares, o que implica em algum tipo de sincronização (e obviamente abre as portas para inúmeros níveis de multiprocessamento). Três métodos são usados para obter sincronização: E/S programada (o mais primitivo: a CPU espera pelo dispositivo de E/S), E/S dirigido por interrupção, onde a CPU comanda a transferência, mas não fica aguardando sua conclusão. Aqui o dispositivo de E/S interrompe a CPU quando acabar. Durante este tempo, a CPU pode realizar outras tarefas. Finalmente, o método de Acesso Direto à memória (DMA), transfere um grande bloco de dados entre o dispositivo (rápido) de E/S e a memória, sem precisar passar pela CPU. Isto requer um controlador de DMA.

CPUS modernas
A seguir algumas descrições de processadores modernos, no estado da arte em 2024:

- CPU Intel Core i9-12900K: 5,5 GHz, 16 núcleos, 24 threads, até 5,5 GHz de clock boost, taxa de ciclo de máquina estimada entre 220 e 275 ciclos por instrução (CPI).
- CPU AMD Ryzen 9 5950X: 4,9 GHz, 16 núcleos, 32 threads, até 5,0 GHz de clock boost, taxa de ciclo de máquina estimada entre 200 e 250 CPI.
- CPU Apple M1 Pro: 3,2 GHz, 10 núcleos (8 núcleos de alto desempenho + 2 núcleos de alta eficiência), 16 threads, até 3,2 GHz de clock boost, taxa de ciclo de máquina estimada entre 150 e 200 CPI.

A taxa de ciclo é uma medida da eficiência ao executar uma instrução: ela indica o número de ciclos de relógio necessários para executar 1 instrução. O clock boost é uma execução acelerada que pode ser invocada pela CPU (depende de temperatura, carga, eficiência energética, etc)

Um núcleo é capaz de executar uma instrução por vez. Cada núcleo possui seus próprios recursos dedicados, como registradores e unidades de execução, permitindo que ele processe tarefas de forma independente. Quanto mais núcleos uma CPU possui ela pode lidar com mais tarefas simultaneamente.

As threads, por outro lado, dividem o tempo de processamento com os núcleos. Elas compartilham os mesmos recursos do núcleo, mas podem alternar a execução de diferentes instruções. Isso permite que um único núcleo execute várias threads ao mesmo tempo, aumentando ainda mais a capacidade multitarefa da CPU.

Uma analogia: Imagine uma cozinha como a CPU e os chefs como os núcleos. Cada chef (núcleo) tem sua própria estação de trabalho (recursos dedicados) e pode preparar um prato (instrução) por vez. As garçonetes (threads) circulam pela cozinha, pegando os pratos prontos dos chefs e entregando-os aos clientes (aplicativos). Quanto mais chefs (núcleos) e garçonetes (threads) a cozinha tiver, mais pratos (tarefas) ela poderá preparar e servir (executar) ao mesmo tempo.

Diferentes arquiteturas

CISC acrônimo de *Complex Instruction Set Computer*. Tem um grande conjunto de instruções, incluindo muitas complexas. Torna o processo de programação mais fácil, já que para muitas tarefas, há uma instrução nativa. A complexidade das instruções torna os circuitos da CPU e da UC mais complicados. A maneira de lidar com isso foi construir a programação em dois níveis. No primeiro, chamado micro-operações, estão apenas as operações muito simples, as únicas efetuadas diretamente pela CPU. Surge uma micromemória, que contém a tradução das micro-operações da operação complexa que está sendo executada. Como tudo na vida, há vantagens (programas menores no nível da máquina) e desvantagens (sobrecarga da micromemória e da micro-programação). Um exemplo desta arquitetura é a série Pentium da Intel, bem como mainframes e supercomputadores. Note-se que os PCs parecem estar migrando para a arquitetura RISC.

RISC Acrônimo de *Reduced Instruction Set Computer*. A idéia é ter o mínimo possível no conjunto de instruções de máquina, traduzindo operações complexas em macros usando operações simples sob controle do programa (e não da máquina como no caso CISC). Exemplos de computadores RISC: smartphones e tablets (processadores ARM), consoles de videogame (Nintendo Switch), Apple MacBook Air, entre outros.
Para encerrar vale dizer que mais recentemente computadores começaram a usar recursos de ambas as abordagens.

Pipeline Como se viu, um computador executa operações em 3 fases: busca, decodificação e execução. Originalmente, as 3 fases ocorriam em série para cada instrução: a instrução x precisava concluir as 3 fases antes de começar o ciclo de instrução $x+1$. Mais modernamente os computadores começaram a usar a técnica de *pipeline*, permitindo que a UC possa começar as fases da $x+1$ enquanto ainda trata a x.

Processamento paralelo Quando o custo do hardware começou a cair, computadores começaram a ser desenhados com várias UCs, várias ULAs e várias memórias. Isto permite o paralelismo na execução de instruções. Agora surgem 4 possibilidades: SISD (fluxo único de instruções e único de dados), SIMD (fluxo único de instruções e múltiplo de dados), MISD (único de dados e múltiplo de instruções) e MIMD (múltiplos ambos). Vale notar que o MISD vale como conceito, mas parece não ter sido nunca implementado. A idéia do processamento paralelo, parece mais fácil como conceito do que na prática. Há aqui um problema de balanceamento de carga: a alocação dinâmica de tarefas para que estas sejam executadas nos diversos processardores disponíveis, mas de forma que todos os processadores sejam utilizados adequadamente. Associado a isto está o problema do escalonamento, em inglês *scaling* (divisão de uma tarefa em sub-tarefas de acordo com o número de processadores disponíveis).

Com o aumento do número de tarefas, cresce exponencialmente o trabalho necessário para coordenar a interação entre elas. Se houver 4 tarefas, há potencialmente 6 pares de tarefas se comunicando. Em 5 tarefas, há 10 pares e em 6 tarefas, 15 pares.

Filas dentro do Sist. Operacional

Imagine-se um sistema operacional multiprogramado despachando tarefas. Como o processador é um só, e existem muitos processos querendo-o usar, formam-se filas. Alem do algoritmo FIFO também conhecido como FCFS (*fist come, first served*), vão ser usados 2 algoritmos distintos:

SJF *Shortest Job First.* Neste esquema, quando o processador vai atender alguém, ele escolhe na fila de espera, aquele que tiver a menor requisição de tempo. Se houver empate, o primeiro será retirado.

RR *Round Robin.* Neste esquema preemptivo, a fila será atendida em ordem, mas com uma quantidade fixa de tempo para cada processo. Após receber este *quantum*, o processo vai para o fim da fila.

Para este problema, haverá um padrão de chegadas de requisições de tempo. A unidade a ser considerada é o segundo. Não haverá nenhum tempo de *overhead* para troca de processos. Irão ser usados para cada processo, dois tempos, assim definidos:

duração do processo tempo de fim - tempo de início + 1

elapsed time tempo de fim - tempo de chegada + 1

Acompanhe no exemplo feito, usando o método **Round Robin, com q=7**

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
H	4	8	4	25	22	22
G	5	9	11	41	31	37
D	7	7	18	24	7	18
I	11	3	26	28	3	18
A	13	4	29	32	4	20
P	15	9	33	80	48	66
X	19	7	42	48	7	30
Z	23	11	49	96	48	74
S	27	5	56	60	5	34
U	32	11	61	100	40	69
B	33	5	68	72	5	40
F	35	2	73	74	2	40
V	38	4	75	78	4	41
W	44	5	81	85	5	42
N	45	10	86	103	18	59

Veja o resultado do mapa do processador, onde cada letra corresponde a 1 seg. Espaços são só para clareza e separam 10 segundos.

```
  HHHHHHH GGGGGGGDDD DDDDHIIIAA AAPPPPPPPG GXXXX
XXXZZ ZZZZZSSSSS UUUUUUUBBB BBFFVVVVPP WWWWWNNNNN
 NNZZZZUUUU NNN
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 16.6 segundos e 40.7 segundos respectivamente.

☞ Para você fazer

Imagine a seguinte situação em um ambiente multiprogramado com o método **Round Robin, com q=7** .

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
P	3	7				
Y	4	4				
N	8	7				
J	9	8				
Q	10	4				
D	11	11				
E	13	8				
L	14	8				
H	18	6				
O	27	4				
R	28	11				
G	31	9				
B	40	8				
C	41	5				
A	44	5				

Para preencher a tabela acima, use o seguinte espaço de tempo

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
100	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150

Responda aqui, qual a média de duração e de elapsed do exercício proposto com o método **Round Robin, com q=7** . (1 casa decimal, por favor)

duração	elapsed

204-76632 - /

CEP - UP - UTFPR - PUCPr - UFPR – 03/06/2024 - 13:31:33.3
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V: 2.02 76649 VICTOR HUGO DOS SANTOS DE CAMA 24CC2204 - 16 _____ / _____ / _____

Executando programas

Um programa é uma unidade de trabalho sendo executado em um computador. Em geral, um programa obtém dados de entrada, processa-os e gera dados de saída. Um detalhe importante é que (graças à arquitetura de Von Neumann) programas são tratados e armazenados na memória como se fossem dados.

A CPU utiliza ciclos de máquina repetitivos para executar as instruções do programa, uma a uma, do início ao fim. Cada ciclo é composto pelas fases de busca, decodificação e execução. Na fase de busca, a unidade de controle manda o sistema copiar no registrador de instruções a próxima instrução. (isto é feito a partir do registrador chamado contador do programa). Ao acabar esta fase o contador é incrementado para apontar para a próxima instrução. A fase de decodificação visa preparar os circuitos necessários para obedecer à instrução, seja ela qual for. Finalmente, a execução, é a implementação da instrução.

Note que as velocidades da CPU e dos dispositivos de E/S são completamente díspares, o que implica em algum tipo de sincronização (e obviamente abre as portas para inúmeros níveis de multiprocessamento). Três métodos são usados para obter sincronização: E/S programada (o mais primitivo: a CPU espera pelo dispositivo de E/S), E/S dirigido por interrupção, onde a CPU comanda a transferência, mas não fica aguardando sua conclusão. Aqui o dispositivo de E/S interrompe a CPU quando acabar. Durante este tempo, a CPU pode realizar outras tarefas. Finalmente, o método de Acesso Direto à memória (DMA), transfere um grande bloco de dados entre o dispositivo (rápido) de E/S e a memória, sem precisar passar pela CPU. Isto requer um controlador de DMA.

CPUS modernas
A seguir algumas descrições de processadores modernos, no estado da arte em 2024:

- CPU Intel Core i9-12900K: 5,5 GHz, 16 núcleos, 24 threads, até 5,5 GHz de clock boost, taxa de ciclo de máquina estimada entre 220 e 275 ciclos por instrução (CPI).
- CPU AMD Ryzen 9 5950X: 4,9 GHz, 16 núcleos, 32 threads, até 5,0 GHz de clock boost, taxa de ciclo de máquina estimada entre 200 e 250 CPI.
- CPU Apple M1 Pro: 3,2 GHz, 10 núcleos (8 núcleos de alto desempenho + 2 núcleos de alta eficiência), 16 threads, até 3,2 GHz de clock boost, taxa de ciclo de máquina estimada entre 150 e 200 CPI.

A taxa de ciclo é uma medida da eficiência ao executar uma instrução: ela indica o número de ciclos de relógio necessários para executar 1 instrução. O clock boost é uma execução acelerada que pode ser invocada pela CPU (depende de temperatura, carga, eficiência energética, etc)

Um núcleo é capaz de executar uma instrução por vez. Cada núcleo possui seus próprios recursos dedicados, como registradores e unidades de execução, permitindo que ele processe tarefas de forma independente. Quanto mais núcleos uma CPU possui ela pode lidar com mais tarefas simultaneamente.

As threads, por outro lado, dividem o tempo de processamento com os núcleos. Elas compartilham os mesmos recursos do núcleo, mas podem alternar a execução de diferentes instruções. Isso permite que um único núcleo execute várias threads ao mesmo tempo, aumentando ainda mais a capacidade multitarefa da CPU.

Uma analogia: Imagine uma cozinha como a CPU e os chefs como os núcleos. Cada chef (núcleo) tem sua própria estação de trabalho (recursos dedicados) e pode preparar um prato (instrução) por vez. As garçonetes (threads) circulam pela cozinha, pegando os pratos prontos dos chefs e entregando-os aos clientes (aplicativos). Quanto mais chefs (núcleos) e garçonetes (threads) a cozinha tiver, mais pratos (tarefas) ela poderá preparar e servir (executar) ao mesmo tempo.

Diferentes arquiteturas

CISC acrônimo de *Complex Instruction Set Computer*. Tem um grande conjunto de instruções, incluindo muitas complexas. Torna o processo de programação mais fácil, já que para muitas tarefas, há uma instrução nativa. A complexidade das instruções torna os circuitos da CPU e da UC mais complicados. A maneira de lidar com isso foi construir a programação em dois níveis. No primeiro, chamado micro-operações, estão apenas as operações muito simples, as únicas efetuadas diretamente pela CPU. Surge uma micromemória, que contém a tradução das micro-operações da operação complexa que está sendo executada. Como tudo na vida, há vantagens (programas menores no nível da máquina) e desvantagens (sobrecarga da micromemória e da micro-programação). Um exemplo desta arquitetura é a série Pentium da Intel, bem como mainframes e supercomputadores. Note-se que os PCs parecem estar migrando para a arquitetura RISC.

RISC Acrônimo de *Reduced Instruction Set Computer*. A idéia é ter o mínimo possível no conjunto de instruções de máquina, traduzindo operações complexas em macros usando operações simples sob controle do programa (e não da máquina como no caso CISC). Exemplos de computadores RISC: smartphones e tablets (processadores ARM), consoles de videogame (Nintendo Switch), Apple MacBook Air, entre outros.
Para encerrar vale dizer que mais recentemente computadores começaram a usar recursos de ambas as abordagens.

Pipeline Como se viu, um computador executa operações em 3 fases: busca, decodificação e execução. Originalmente, as 3 fases ocorriam em série para cada instrução: a instrução x precisava concluir as 3 fases antes de começar o ciclo de instrução $x+1$. Mais modernamente os computadores começaram a usar a técnica de *pipeline*, permitindo que a UC possa começar as fases da $x+1$ enquanto ainda trata a x.

Processamento paralelo Quando o custo do hardware começou a cair, computadores começaram a ser desenhados com várias UCs, várias ULAs e várias memórias. Isto permite o paralelismo na execução de instruções. Agora surgem 4 possibilidades: SISD (fluxo único de instruções e único de dados), SIMD (fluxo único de instruções e múltiplo de dados), MISD (único de dados e múltiplo de instruções) e MIMD (múltiplos ambos). Vale notar que o MISD vale como conceito, mas parece não ter sido nunca implementado. A idéia do processamento paralelo, parece mais fácil como conceito do que na prática. Há aqui um problema de balanceamento de carga: a alocação dinâmica de tarefas para que estas sejam executadas nos diversos processardores disponíveis, mas de forma que todos os processadores sejam utilizados adequadamente. Associado a isto está o problema do escalonamento, em inglês *scaling* (divisão de uma tarefa em sub-tarefas de acordo com o número de processadores disponíveis).

Com o aumento do número de tarefas, cresce exponencialmente o trabalho necessário para coordenar a interação entre elas. Se houver 4 tarefas, há potencialmente 6 pares de tarefas se comunicando. Em 5 tarefas, há 10 pares e em 6 tarefas, 15 pares.

Filas dentro do Sist. Operacional

Imagine-se um sistema operacional multiprogramado despachando tarefas. Como o processador é um só, e existem muitos processos querendo-o usar, formam-se filas. Alem do algoritmo FIFO também conhecido como FCFS (*fist come, first served*), vão ser usados 2 algoritmos distintos:

SJF *Shortest Job First.* Neste esquema, quando o processador vai atender alguém, ele escolhe na fila de espera, aquele que tiver a menor requisição de tempo. Se houver empate, o primeiro será retirado.

RR *Round Robin.* Neste esquema preemptivo, a fila será atendida em ordem, mas com uma quantidade fixa de tempo para cada processo. Após receber este *quantum*, o processo vai para o fim da fila.

Para este problema, haverá um padrão de chegadas de requisições de tempo. A unidade a ser considerada é o segundo. Não haverá nenhum tempo de *overhead* para troca de processos. Irão ser usados para cada processo, dois tempos, assim definidos:

duração do processo tempo de fim - tempo de início + 1

elapsed time tempo de fim - tempo de chegada + 1

Acompanhe no exemplo feito, usando o método **Round Robin, com q=7**

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
E	3	3	3	5	3	3
R	5	8	6	30	25	26
I	6	7	13	19	7	14
M	8	3	20	22	3	15
O	9	10	23	61	39	53
Y	23	11	31	81	51	59
W	27	11	38	99	62	73
L	28	10	45	109	65	82
H	29	7	52	58	7	30
T	34	2	62	63	2	30
D	36	9	64	111	48	76
X	37	10	71	114	44	78
Z	41	10	82	117	36	77
U	44	11	89	121	33	78
Q	45	8	100	122	23	78

Veja o resultado do mapa do processador, onde cada letra corresponde a 1 seg. Espaços são só para clareza e separam 10 segundos.

```
  EEERRRRR RRIIIIIIIM MMOOOOOOOR YYYYYYYWWW WWWWL
LLLLL LHHHHHHHOO OTTDDDDDDD XXXXXXXYYY YZZZZZZZUU
 UUUUUWWWWQ QQQQQQLLLD DXXXZZZUUU UQ
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 29.9 segundos e 51.5 segundos respectivamente.

☞ Para você fazer

Imagine a seguinte situação em um ambiente multiprogramado com o método **Round Robin, com q=7** .

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
J	1	9				
F	3	10				
Q	4	6				
N	14	7				
O	17	2				
D	19	9				
Y	23	9				
W	24	7				
V	26	5				
H	28	3				
K	33	11				
G	34	9				
P	36	5				
T	37	5				
X	41	8				

Para preencher a tabela acima, use o seguinte espaço de tempo

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
100	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150

Responda aqui, qual a média de duração e de elapsed do exercício proposto com o método **Round Robin, com q=7** . (1 casa decimal, por favor)

duração	elapsed

204-76649 - /

CEP - UP - UTFPR - PUCPr - UFPR –
03/06/2024 - 13:31:33.3
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V:
2.02 76656 VITOR EDUARDO DE GOES
FONTES 24CC2204 - 17 _____ / _____ / _____

Executando programas

Um programa é uma unidade de trabalho sendo executado em um computador. Em geral, um programa obtém dados de entrada, processa-os e gera dados de saída. Um detalhe importante é que (graças à arquitetura de Von Neumann) programas são tratados e armazenados na memória como se fossem dados.

A CPU utiliza ciclos de máquina repetitivos para executar as instruções do programa, uma a uma, do início ao fim. Cada ciclo é composto pelas fases de busca, decodificação e execução. Na fase de busca, a unidade de controle manda o sistema copiar no registrador de instruções a próxima instrução. (isto é feito a partir do registrador chamado contador do programa). Ao acabar esta fase o contador é incrementado para apontar para a próxima instrução. A fase de decodificação visa preparar os circuitos necessários para obedecer à instrução, seja ela qual for. Finalmente, a execução, é a implementação da instrução.

Note que as velocidades da CPU e dos dispositivos de E/S são completamente díspares, o que implica em algum tipo de sincronização (e obviamente abre as portas para inúmeros níveis de multiprocessamento). Três métodos são usados para obter sincronização: E/S programada (o mais primitivo: a CPU espera pelo dispositivo de E/S), E/S dirigido por interrupção, onde a CPU comanda a transferência, mas não fica aguardando sua conclusão. Aqui o dispositivo de E/S interrompe a CPU quando acabar. Durante este tempo, a CPU pode realizar outras tarefas. Finalmente, o método de Acesso Direto à memória (DMA), transfere um grande bloco de dados entre o dispositivo (rápido) de E/S e a memória, sem precisar passar pela CPU. Isto requer um controlador de DMA.

CPUS modernas
A seguir algumas descrições de processadores modernos, no estado da arte em 2024:

- CPU Intel Core i9-12900K: 5,5 GHz, 16 núcleos, 24 threads, até 5,5 GHz de clock boost, taxa de ciclo de máquina estimada entre 220 e 275 ciclos por instrução (CPI).
- CPU AMD Ryzen 9 5950X: 4,9 GHz, 16 núcleos, 32 threads, até 5,0 GHz de clock boost, taxa de ciclo de máquina estimada entre 200 e 250 CPI.
- CPU Apple M1 Pro: 3,2 GHz, 10 núcleos (8 núcleos de alto desempenho + 2 núcleos de alta eficiência), 16 threads, até 3,2 GHz de clock boost, taxa de ciclo de máquina estimada entre 150 e 200 CPI.

A taxa de ciclo é uma medida da eficiência ao executar uma instrução: ela indica o número de ciclos de relógio necessários para executar 1 instrução. O clock boost é uma execução acelerada que pode ser invocada pela CPU (depende de temperatura, carga, eficiência energética, etc)

Um núcleo é capaz de executar uma instrução por vez. Cada núcleo possui seus próprios recursos dedicados, como registradores e unidades de execução, permitindo que ele processe tarefas de forma independente. Quanto mais núcleos uma CPU possui ela pode lidar com mais tarefas simultaneamente.

As threads, por outro lado, dividem o tempo de processamento com os núcleos. Elas compartilham os mesmos recursos do núcleo, mas podem alternar a execução de diferentes instruções. Isso permite que um único núcleo execute várias threads ao mesmo tempo, aumentando ainda mais a capacidade multitarefa da CPU.

Uma analogia: Imagine uma cozinha como a CPU e os chefs como os núcleos. Cada chef (núcleo) tem sua própria estação de trabalho (recursos dedicados) e pode preparar um prato (instrução) por vez. As garçonetes (threads) circulam pela cozinha, pegando os pratos prontos dos chefs e entregando-os aos clientes (aplicativos). Quanto mais chefs (núcleos) e garçonetes (threads) a cozinha tiver, mais pratos (tarefas) ela poderá preparar e servir (executar) ao mesmo tempo.

Diferentes arquiteturas

CISC acrônimo de *Complex Instruction Set Computer*. Tem um grande conjunto de instruções, incluindo muitas complexas. Torna o processo de programação mais fácil, já que para muitas tarefas, há uma instrução nativa. A complexidade das instruções torna os circuitos da CPU e da UC mais complicados. A maneira de lidar com isso foi construir a programação em dois níveis. No primeiro, chamado micro-operações, estão apenas as operações muito simples, as únicas efetuadas diretamente pela CPU. Surge uma micromemória, que contém a tradução das micro-operações da operação complexa que está sendo executada. Como tudo na vida, há vantagens (programas menores no nível da máquina) e desvantagens (sobrecarga da micromemória e da micro-programação). Um exemplo desta arquitetura é a série Pentium da Intel, bem como mainframes e supercomputadores. Note-se que os PCs parecem estar migrando para a arquitetura RISC.

RISC Acrônimo de *Reduced Instruction Set Computer*. A idéia é ter o mínimo possível no conjunto de instruções de máquina, traduzindo operações complexas em macros usando operações simples sob controle do programa (e não da máquina como no caso CISC). Exemplos de computadores RISC: smartphones e tablets (processadores ARM), consoles de videogame (Nintendo Switch), Apple MacBook Air, entre outros.
Para encerrar vale dizer que mais recentemente computadores começaram a usar recursos de ambas as abordagens.

Pipeline Como se viu, um computador executa operações em 3 fases: busca, decodificação e execução. Originalmente, as 3 fases ocorriam em série para cada instrução: a instrução x precisava concluir as 3 fases antes de começar o ciclo de instrução $x + 1$. Mais modernamente os computadores começaram a usar a técnica de *pipeline*, permitindo que a UC possa começar as fases da $x + 1$ enquanto ainda trata a x.

Processamento paralelo Quando o custo do hardware começou a cair, computadores começaram a ser desenhados com várias UCs, várias ULAs e várias memórias. Isto permite o paralelismo na execução de instruções. Agora surgem 4 possibilidades: SISD (fluxo único de instruções e único de dados), SIMD (fluxo único de instruções e múltiplo de dados), MISD (único de dados e múltiplo de instruções) e MIMD (múltiplos ambos). Vale notar que o MISD vale como conceito, mas parece não ter sido nunca implementado. A idéia do processamento paralelo, parece mais fácil como conceito do que na prática. Há aqui um problema de balanceamento de carga: a alocação dinâmica de tarefas para que estas sejam executadas nos diversos processardores disponíveis, mas de forma que todos os processadores sejam utilizados adequadamente. Associado a isto está o problema do escalonamento, em inglês *scaling* (divisão de uma tarefa em sub-tarefas de acordo com o número de processadores disponíveis).

Com o aumento do número de tarefas, cresce exponencialmente o trabalho necessário para coordenar a interação entre elas. Se houver 4 tarefas, há potencialmente 6 pares de tarefas se comunicando. Em 5 tarefas, há 10 pares e em 6 tarefas, 15 pares.

Filas dentro do Sist. Operacional

Imagine-se um sistema operacional multiprogramado despachando tarefas. Como o processador é um só, e existem muitos processos querendo-o usar, formam-se filas. Alem do algoritmo FIFO também conhecido como FCFS (*fist come, first served*), vão ser usados 2 algoritmos distintos:

SJF *Shortest Job First*. Neste esquema, quando o processador vai atender alguém, ele escolhe na fila de espera, aquele que tiver a menor requisição de tempo. Se houver empate, o primeiro será retirado.

RR *Round Robin*. Neste esquema preemptivo, a fila será atendida em ordem, mas com uma quantidade fixa de tempo para cada processo. Após receber este *quantum*, o processo vai para o fim da fila.

Para este problema, haverá um padrão de chegadas de requisições de tempo. A unidade a ser considerada é o segundo. Não haverá nenhum tempo de *overhead* para troca de processos. Irão ser usados para cada processo, dois tempos, assim definidos:

duração do processo tempo de fim - tempo de início + 1

elapsed time tempo de fim - tempo de chegada + 1

Acompanhe no exemplo feito, usando o método **Shortest Job First**

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
I	1	8	1	8	8	8
G	4	7	17	23	7	20
Y	6	4	9	12	4	7
R	11	11	71	81	11	71
B	12	4	13	16	4	5
K	24	7	24	30	7	7
E	25	8	34	41	8	17
P	27	3	31	33	3	7
C	29	11	82	92	11	64
Q	35	8	63	70	8	36
H	36	6	42	47	6	12
M	37	6	50	55	6	19
T	39	7	56	62	7	24
A	40	11	93	103	11	64
J	44	2	48	49	2	6

Veja o resultado do mapa do processador, onde cada letra corresponde a 1 seg. Espaços são só para clareza e separam 10 segundos.

```
IIIIIIIIYY YYBBBBGGGG GGGKKKKKKK PPPEEEEEEE EHHHH
HHJJM MMMMMTTTTT TTQQQQQQQQ RRRRRRRRR RCCCCCCCCC
 CCAAAAAAAA AAA
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 6.9 segundos e 24.5 segundos respectivamente.

☞ Para você fazer

Imagine a seguinte situação em um ambiente multiprogramado com o método **Shortest Job First** .

proc	T cheg	Ped.	T in.	T fim	Dur	Elap
L	5	11				
Q	6	11				
J	9	8				
O	18	10				
D	21	11				
W	22	11				
K	23	3				
Y	27	4				
I	29	6				
H	30	11				
Z	40	7				
F	41	6				
X	42	6				
N	44	2				
A	45	2				

Para preencher a tabela acima, use o seguinte espaço de tempo

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
100	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150

Responda aqui, qual a média de duração e de elapsed do exercício proposto com o método **Shortest Job First** . (1 casa decimal, por favor)

duração	elapsed