CEP - UP - UTFPR - PUCPr - UFPR – 10/06/2024 - 08:55:35.1
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V: 2.02 76201 ANA GLORIA DE SOUZA BRITO (76201);
HENRIQUE NOGUEIRA RIBEIRO (76368);
24CC1204 - 1 27/jun limite 1. avaliacao _____ / / _____

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
Y	7	10	7	41	35	35
M	8	4	10	19	10	12
F	10	9	16	62	47	53
B	13	9	20	71	52	59
D	18	9	26	88	63	71
L	19	9	32	100	69	82
Z	23	7	38	101	64	79
H	26	5	42	79	38	54
R	27	9	45	107	63	81
W	28	3	48	50	3	23
P	29	10	54	118	65	90
S	31	8	57	112	56	82
O	32	8	63	114	52	83
A	41	10	75	119	45	79
U	45	3	80	82	3	38

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

```
     YYYM MMYYYFFFMB BBYYYDDDFF FLLLBBBZZZ YHHHR
RRWWW DDDPPPSSSF FFOOOLLLBB BZZZAAAHHU UURRRDDDPP
 PSSSOOOLLL ZAAARRRPPP SSOOAAAPA
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 44.3 segundos e 61.4 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
V	2	8				
F	3	7				
H	4	6				
G	6	9				
Q	7	4				
C	10	7				
M	21	5				
E	23	6				
L	28	9				
U	31	7				
K	34	6				
O	36	2				
W	41	10				
P	43	9				
T	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
101	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

204-76201 - n /im

CEP - UP - UTFPR - PUCPr - UFPR – 10/06/2024 - 08:55:35.1
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V: 2.02 76463 PEDRO HENRIQUE STAATS (76463);
SARAH FERNANDES CHELIGA (76470);
24CC1204 - 2 27/jun limite 1. avaliacao _____ / / _____

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
V	1	3	1	3	3	3
Z	2	2	4	5	2	4
D	5	7	6	12	7	8
O	11	5	13	17	5	7
A	12	8	32	39	8	28
L	13	10	76	85	10	73
W	15	4	18	21	4	7
F	16	6	22	27	6	12
B	19	8	40	47	8	29
I	23	8	51	58	8	36
H	27	4	28	31	4	5
G	29	8	59	66	8	38
N	30	11	86	96	11	67
U	32	9	67	75	9	44
Q	41	3	48	50	3	10

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

```
VVVZZDDDDD DDOOOOOWWW WFFFFFFHHH HAAAAAAAAB BBBBB
BBQQQ IIIIIIIIGG GGGGGGUUUU UUUUULLLLL LLLLLNNNNN
 NNNNNN
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 6.4 segundos e 24.7 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	5	8				
N	7	3				
S	9	7				
A	12	7				
Q	21	5				
W	23	2				
Y	24	9				
R	26	6				
I	27	5				
U	34	11				
X	37	3				
H	39	6				
O	41	11				
D	43	4				
M	44	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
101	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

204-76463 - n /im

CEP - UP - UTFPR - PUCPr - UFPR – 10/06/2024 - 08:55:35.1
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V: 2.02 76494 HENRIQUE RANKEL CRENCHIGLOV(76494);
PEDRO CAPOTE DA CONCEICAO S(76449);
24CC1204 - 3 27/jun limite 1. avaliacao _____ / / _____

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
L	4	5	4	8	5	5
J	6	8	9	37	29	32
B	8	10	16	56	41	49
O	10	7	23	29	7	20
F	13	7	30	36	7	24
S	16	8	38	93	56	78
Y	21	10	45	96	52	76
W	22	2	52	53	2	32
T	30	10	57	99	43	70
Q	33	3	64	66	3	34
V	34	6	67	72	6	39
A	40	5	73	77	5	38
P	41	11	78	103	26	63
H	42	2	85	86	2	45
U	44	6	87	92	6	49

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

```
LLLLLJJ JJJJJBBBBB BBOOOOOOOF FFFFFFJSSS SSSSY
YYYYY YWWBBBTTTT TTTQQQVVVV VVAAAAAPPP PPPPHHUUUU
 UUSYYYTTTP PPP
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 19.3 segundos e 43.6 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
K	1	9				
P	4	5				
T	7	9				
X	10	4				
H	11	3				
I	13	10				
M	19	8				
U	21	2				
R	25	6				
L	26	11				
O	30	4				
Z	37	5				
F	39	4				
J	40	8				
W	44	7				

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
101	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

204-76494 - n /im

CEP - UP - UTFPR - PUCPr - UFPR – 10/06/2024 - 08:55:35.1
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V: 2.02 76263 DAVI HONORIO ROSA (76263); LORENZO QUIRINO DE SALLES (76399); 24CC1204 - 4 27/jun limite 1. avaliacao _____ / / _____

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	1	8	1	8	8	8
J	3	7	16	22	7	20
V	7	2	9	10	2	4
P	9	5	11	15	5	7
W	10	10	59	68	10	59
L	11	9	50	58	9	48
C	17	5	29	33	5	17
B	20	4	23	26	4	7
X	24	2	27	28	2	5
I	28	5	37	41	5	14
Z	32	10	69	78	10	47
D	33	3	34	36	3	4
O	36	5	45	49	5	14
K	41	3	42	44	3	4
Y	42	11	79	89	11	48

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

```
GGGGGGGGVV PPPPPJJJJJ JJBBBBXXCC CCCDDDIIII IKKKO
OOOOL LLLLLLLLWW WWWWWWWWZZ ZZZZZZZZYY YYYYYYYYY
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 5.9 segundos e 20.4 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
W	3	6				
P	9	10				
X	12	4				
K	13	4				
C	14	5				
Y	16	9				
L	17	3				
Q	27	11				
R	29	3				
O	30	3				
Z	32	5				
S	34	5				
D	40	9				
F	41	9				
M	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
101	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

204-76263 - n /im

CEP - UP - UTFPR - PUCPr - UFPR –
10/06/2024 - 08:55:35.1
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V:
2.02 76487 WILLIAN IELEM PAVOSKI (76487);
LUAN TEIXEIRA DIBE (76401);
24CC1204 - 5 27/jun limite 1. avaliacao _____ / / _____

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	3	4	3	6	4	4
Y	4	10	7	42	36	39
K	6	5	14	18	5	13
H	9	7	19	25	7	17
Q	10	9	26	70	45	61
R	11	8	33	79	47	69
N	17	11	43	96	54	80
O	20	8	50	97	48	78
M	25	10	57	100	44	76
E	26	5	64	68	5	43
D	33	2	71	72	2	40
A	36	6	73	78	6	43
C	41	2	80	81	2	41
P	42	4	82	85	4	44
Z	44	10	86	103	18	60

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

```
 IIIIYYYY YYYKKKKKHH HHHHHQQQQQ QQRRRRRRRY YYNNN
NNNNO OOOOOOMMMM MMMEEEEEQQ DDAAAAAARC CPPPPZZZZZ
 ZZNNNNOMMM ZZZ
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 21.8 segundos e 47.2 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
I	6	8				
E	11	4				
B	15	4				
U	16	8				
V	21	6				
T	23	4				
D	24	10				
Y	28	8				
F	33	11				
A	35	11				
X	37	6				
Q	39	5				
W	40	10				
L	42	4				
N	44	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
101	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 –
10/06/2024 - 08:55:35.1
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V:
2.02 76351 HENRI KIM PINHEIRO
RODRIGUE(76351);
ANDRE SANCHES (76218);
24CC1204 - 6 27/jun limite 1. avaliacao _____ /
/ _____

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
S	1	3	1	3	3	3
F	7	7	7	19	13	13
U	10	10	13	80	68	71
Z	11	11	16	85	70	75
O	13	5	20	39	20	27
V	14	7	23	78	56	65
K	21	9	32	88	57	68
T	22	6	35	66	32	45
R	24	8	40	95	56	72
C	28	4	46	79	34	52
G	29	3	52	54	3	26
P	30	6	55	83	29	54
X	38	7	67	96	30	59
Q	42	5	70	93	24	52
H	43	2	76	77	2	35

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

```
SSS   FFFF FFUUUZZZFO OOVVVUUUZZ ZKKKTTTOOR RRVVV
CCCUU UGGGPPPZZZ KKKTTTXXXQ QQRRRHHVCU PPPZZKKKXX
 XQQRRX
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 33.1 segundos e 47.8 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
G	3	6				
E	5	7				
Y	6	10				
L	8	9				
I	11	7				
U	14	3				
X	17	8				
W	23	6				
V	28	8				
T	30	5				
R	37	2				
B	39	7				
M	41	7				
J	43	9				
Q	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
101	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

204-76351 - n /im

CEP - UP - UTFPR - PUCPr - UFPR –
10/06/2024 - 08:55:35.1
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V: 2.02 76313 GABRIEL BORDIGNON (76313); CAUA DOS SANTOS COSTA (76232);
24CC1204 - 7 27/jun limite 1. avaliacao _____ / / _____

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
K	6	4	6	9	4	4
H	8	6	10	15	6	8
M	11	9	29	37	9	27
I	13	5	16	20	5	8
C	16	10	77	86	10	71
Q	17	9	68	76	9	60
X	21	8	21	28	8	8
Z	26	11	97	107	11	82
U	35	7	53	59	7	25
G	37	6	38	43	6	7
P	39	8	60	67	8	29
E	40	4	49	52	4	13
V	42	10	87	96	10	55
T	44	2	44	45	2	2
D	45	3	46	48	3	4

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

```
     KKKKH HHHHHIIIII XXXXXXXXMM MMMMMMMGGG GGGTT
DDDEE EEUUUUUUUP PPPPPPQQQ QQQQQQCCCC CCCCCVVVV
 VVVVVVZZZZ ZZZZZZZ
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 6.8 segundos e 26.9 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
Q	2	10				
A	7	5				
J	8	11				
K	12	10				
Y	14	3				
L	18	9				
C	21	10				
R	22	5				
G	24	8				
V	25	4				
Z	28	10				
H	32	5				
M	40	11				
U	42	7				
S	44	3				

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
101	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

204-76313 - n /im

CEP - UP - UTFPR - PUCPr - UFPR – 10/06/2024 - 08:55:35.1
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V: 2.02 76456 PEDRO HENRIQUE DE SOUZA (76456);
JOAO FRANCISCO LEONEL DE SO(76375);
24CC1204 - 8 27/jun limite 1. avaliacao _____ / / _____

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
U	5	8	5	18	14	14
D	8	11	11	68	58	61
W	9	3	14	16	3	8
X	13	3	19	21	3	9
P	17	4	25	46	22	30
C	18	11	28	101	74	84
H	20	5	31	57	27	38
M	23	10	34	106	73	84
S	25	3	40	42	3	18
L	26	11	43	108	66	83
B	28	7	47	98	52	71
O	29	7	50	99	50	71
J	35	10	58	109	52	75
Q	36	7	61	105	45	70
A	40	5	69	94	26	55

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

```
    UUUUUU DDDWWWUUXX XDDDPPPCCC HHHMMMDDDS SSLLL
PBBBO OOCCCHHJJJ QQQMMMDDAA ALLLBBBOOO CCCJJJQQQM
 MMAALLLBOC CJJJQMLLJ
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 37.9 segundos e 51.4 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
C	2	2				
X	5	10				
E	11	6				
S	12	5				
J	14	4				
T	15	4				
N	18	9				
L	19	9				
M	27	4				
I	28	3				
V	29	3				
R	37	3				
D	42	11				
Y	43	7				
B	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
101	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 – 10/06/2024 - 08:55:35.1
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V: 2.02 76294 FABRICIO JAWORSKI FERREIRA (76294);
ENZO KAUAN CARVALHO (76287);
24CC1204 - 9 27/jun limite 1. avaliacao _____ / / _____

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
R	3	7	3	9	7	7
B	4	9	27	35	9	32
Z	6	7	10	16	7	11
N	8	8	17	24	8	17
X	18	11	93	103	11	86
P	25	2	25	26	2	2
U	28	9	57	65	9	38
Y	30	2	36	37	2	8
F	32	2	38	39	2	8
G	33	9	66	74	9	42
D	34	8	49	56	8	23
W	36	9	75	83	9	48
M	40	7	40	46	7	7
K	41	9	84	92	9	52
V	44	2	47	48	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.

```
 RRRRRRRZ ZZZZZZNNNN NNNNPPBBBB BBBBBYYFFM MMMMM
MVVDD DDDDDDUUUU UUUUUGGGGG GGGGWWWWWW WWWKKKKKKK
 KKXXXXXXXX XXX
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 6.7 segundos e 25.7 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	1	4				
S	2	9				
A	3	7				
X	4	10				
B	5	7				
K	7	5				
H	8	5				
O	9	4				
R	11	9				
Y	16	5				
E	18	11				
C	23	7				
U	24	3				
W	39	5				
P	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
101	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

204-76294 - n /im

CEP - UP - UTFPR - PUCPr - UFPR – 10/06/2024 - 08:55:35.1
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V: 2.02 76432 PAULO GUSTAVO TORRES (76432);
LANA ALESSANDRA GALEGO VEIG(76382);
24CC1204 - 10 27/jun limite 1. avaliacao _____ / / _____

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
E	2	2	2	3	2	2
G	4	4	4	7	4	4
U	5	2	8	9	2	5
W	6	7	10	16	7	11
S	8	7	26	32	7	25
Z	10	11	75	85	11	76
B	11	8	58	65	8	55
H	14	5	21	25	5	12
A	15	4	17	20	4	6
P	27	6	37	42	6	16
C	30	4	33	36	4	7
L	31	9	66	74	9	44
I	39	4	43	46	4	8
R	43	4	47	50	4	8
F	45	7	51	57	7	13

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

```
EEGGGGUUW WWWWWWAAAA HHHHHSSSSS SSCCCCPPPP PPIII
IRRRR FFFFFFFBBB BBBBBLLLLL LLLLZZZZZZ ZZZZZ
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 5.6 segundos e 19.5 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
S	2	4				
U	4	11				
T	7	8				
Z	10	9				
J	11	10				
L	12	8				
I	18	6				
F	19	3				
M	20	6				
D	21	9				
C	24	2				
W	26	4				
N	34	9				
G	40	2				
A	42	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
101	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

204-76432 - n /im

CEP - UP - UTFPR - PUCPr - UFPR – 10/06/2024 - 08:55:35.1
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V: 2.02 76256 CLEVERSON LUIZ VOITACH JUNI(76256);
GUILHERME LUCIANO DE MELO (76344);
24CC1204 - 11 27/jun limite 1. avaliacao _____ / / _____

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
U	3	6	3	11	9	9
C	5	10	6	35	30	31
D	6	2	12	13	2	8
Q	10	3	17	19	3	10
N	16	11	20	72	53	57
Y	20	3	26	28	3	9
Z	23	5	32	58	27	36
T	26	5	36	60	25	35
R	27	4	39	64	26	38
L	29	9	42	88	47	60
E	32	5	48	74	27	43
B	33	10	51	95	45	63
V	34	7	54	92	39	59
O	41	8	61	94	34	54
P	43	5	65	85	21	43

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

```
 UUUCCCUU UDDCCCQQQN NNCCCYYYNN NZZZCTTTRR RLLLN
NNEEE BBBVVVZZTT OOORPPPLLL NNEEBBBVVV OOOPPLLLBB
 BVOOB
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 26.1 segundos e 37.0 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
Q	5	2				
S	8	5				
X	10	4				
P	11	9				
A	13	11				
E	16	7				
N	22	7				
L	24	2				
W	30	4				
K	31	3				
I	38	4				
U	40	10				
J	42	10				
Z	43	6				
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
101	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 – 10/06/2024 - 08:55:35.1
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V: 2.02 76249 CAUE DAS NEVES MACHADO (76249);
FELIPE FERNANDES DOS SANTOS(76306);
24CC1204 - 12 27/jun limite 1. avaliacao _____ / / _____

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	6	3	14	12	12
Z	4	11	6	60	55	57
D	5	6	9	32	24	28
S	6	4	15	36	22	31
K	9	8	21	68	48	60
F	10	8	24	73	50	64
L	11	7	27	76	50	66
Y	16	3	33	35	3	20
V	27	5	46	75	30	49
C	32	2	52	53	2	22
B	37	3	54	56	3	20
U	39	2	57	58	2	20
T	40	11	61	94	34	55
G	42	9	64	91	28	50
W	44	7	69	92	24	49

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

```
HHHZZZDD DHHHSSSZZZ KKKFFFLLLD DDYYYSZZZK KKFFF
VVVLL LCCBBBUUZZ TTTGGGKKWW WFFVVLTTTG GGWWWTTTGG
 GWTT
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 25.8 segundos e 40.2 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
N	6	2				
L	7	7				
A	8	4				
D	19	2				
E	21	9				
Q	23	5				
R	29	4				
K	34	3				
G	35	4				
T	36	7				
P	37	11				
W	38	5				
Z	41	7				
U	43	11				
B	44	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
101	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

204-76249 - n /im

CEP - UP - UTFPR - PUCPr - UFPR – 10/06/2024 - 08:55:35.1
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V: 2.02 76418 MURILO DIRCKSEN CAMPOS (76418);
GUILHERME CAVALER KUSTER (76337);
24CC1204 - 13 27/jun limite 1. avaliacao _____ / / _____

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
L	1	9	1	9	9	9
N	10	10	10	37	28	28
O	14	10	16	74	59	61
J	15	11	19	82	64	68
Q	19	3	28	30	3	12
Z	24	9	34	86	53	63
A	26	8	38	97	60	72
H	29	7	44	107	64	79
V	34	4	50	83	34	50
X	37	7	56	108	53	72
T	38	7	59	109	51	72
B	40	6	62	95	34	56
M	41	11	68	120	53	80
E	42	10	71	121	51	80
F	44	9	75	118	44	75

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

```
LLLLLLLLLN NNNNNOOOJJ JNNNOOOQQQ JJJZZZNAAA OOOHH
HJJJV VVZZZXXXTT TBBBAAAMMM EEEOFFFHHH JJVZZZXXXT
 TTBBBAAMMM EEEFFFHXTM MMEEEFFFMM E
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 44.0 segundos e 58.5 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
C	5	5				
I	7	8				
U	8	11				
Y	10	6				
E	13	4				
J	21	5				
O	25	6				
L	28	8				
H	31	3				
D	35	8				
T	38	8				
N	41	5				
K	43	9				
B	44	11				
Z	45	7				

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
101	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

204-76418 - n /im

CEP - UP - UTFPR - PUCPr - UFPR –
10/06/2024 - 08:55:35.1
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V:
2.02 76225 CASSIO COELHO PAZ (76225);
DORA DOS SANTOS PELLANDA (76270);
24CC1204 - 14 27/jun limite 1. avaliacao _____ /
/ _____

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
A	1	8	1	19	19	19
I	3	4	8	11	4	9
V	7	8	12	32	21	26
E	8	5	20	24	5	17
Q	11	5	25	29	5	19
Z	14	2	30	31	2	18
P	19	8	33	62	30	44
R	20	2	40	41	2	22
X	22	4	42	45	4	24
D	26	9	46	84	39	59
K	35	4	53	56	4	22
G	38	5	57	61	5	24
O	40	8	63	85	23	46
T	41	9	70	87	18	47
M	43	6	77	82	6	40

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

```
AAAAAAAIII IVVVVVVVAE EEEEQQQQQZ ZVPPPPPPPR RXXXX
DDDDD DDKKKKGGGG GPOOOOOOOT TTTTTTMMMM MMDDOTT
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 12.5 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
F	3	2				
L	6	6				
K	11	2				
Y	16	11				
Z	21	2				
V	25	4				
B	26	9				
Q	29	10				
W	30	6				
G	31	11				
X	32	9				
T	38	4				
U	39	6				
S	42	10				
R	45	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
101	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

204-76225 - n /im

CEP - UP - UTFPR - PUCPr - UFPR – 10/06/2024 - 08:55:35.1
Prof Dr P Kantek (pkantek@gmail.com)
Arquitetura de computadores II VIVXq05c, V: 2.02 76320 GEOVANNI TEIXEIRA ARAUJO (76320);
OLAVO AUGUSTO ROSSI (76425);
24CC1204 - 15 27/jun limite 1. avaliacao _____ / / _____

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
J	1	5	1	5	5	5
C	3	9	21	29	9	27
F	5	4	6	9	4	5
W	8	7	10	16	7	9
L	16	4	17	20	4	5
T	22	5	30	34	5	13
P	24	5	35	39	5	16
D	26	11	70	80	11	55
K	29	11	81	91	11	63
G	31	7	46	52	7	22
V	32	6	40	45	6	14
B	33	11	92	102	11	70
Z	37	11	103	113	11	77
X	40	10	60	69	10	30
I	45	7	53	59	7	15

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

```
JJJJJFFFFW WWWWWLLLL CCCCCCCCCT TTTTPPPPPV VVVVV
GGGGG GGIIIIIIIX XXXXXXXXD DDDDDDDDD KKKKKKKKK
 KBBBBBBBBB BBZZZZZZZZ ZZZ
```

O que se quer saber são as médias de duração e de elapsed. No exemplo acima, elas são de 7.5 segundos e 28.4 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
R	1	9				
U	2	7				
N	6	9				
V	11	6				
X	17	6				
Q	19	3				
H	27	11				
A	28	8				
P	30	2				
C	31	8				
Z	33	5				
S	35	9				
W	37	3				
B	38	9				
O	44	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
101	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

204-76320 - n /im