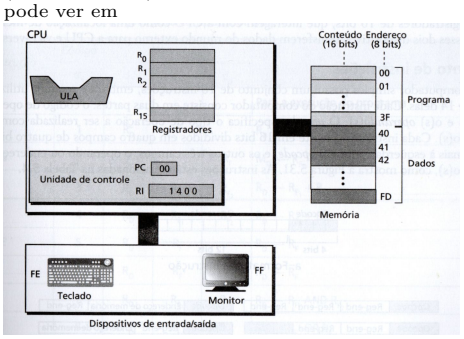
CEP - UP - UTFPR - PUCPr - UFPR –
10/06/2024 - 08:33:04.1
Prof Dr P Kantek (pkantek@gmail.com)
Um computador simples VIVXq06a, V: 1.31
76506 CARLOS EDUARDO ANTAO 24CC2205 -
1 27/jun limite 1. avaliacao _____ / _____ /

Um computador simples

Vamos projetar um computador simples, para estudar de perto a sua arquitetura e seu processamento de instruções. O fato de ser um computador teórico (portanto irreal) não atrapalha. Projetado como se pode ver em



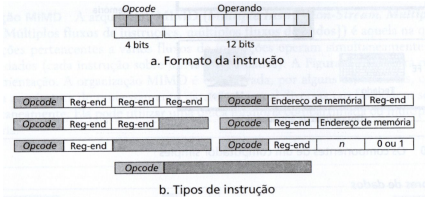
o computador terá 3 componentes: CPU, memória e subsistema de E/S. A CPU é dividida em 3 seções: registradores de dados, unidade aritmético-lógica e unidade de controle.

Existem 16 registradores de dados de 16 bits cada, e com endereços hexadecimais 0,1,...,F, conhecidos como $R_0, R_1, ..., R_{15}$. A unidade de controle controla as operações da ULA e os acessos à memória e ao subsistema de E/S. Tem 2 registradores dedicados: o PC (*Program Counter*) que aponta para o endereço de memória que contém a próxima instrução e o RI (*registrador de instrução*) mantém a instrução que está sendo executada no ciclo corrente. Após cada ciclo, o PC é incrementado (quase sempre) em 1 e o RI é atualizado com a próxima instrução.

A memória principal tem 256 localizações de 16 bits cada. Seus endereços binários variam entre 00000000_2 e 11111101_2 ou entre 00_{16} e FD_{16}. A memória contém dados e instruções. As primeiras 64 localizações (de 00 a $3F_{16}$ são dedicadas a instruções que são armazenadas em posições consecutivas. Já a área de 40 a FD_{16} são utilizadas para armazenar dados.

Já o subsistema de E/S é bastante primitivo, consistindo de um teclado e um vídeo. Assume-se que o teclado usa a posição de memória FE_{16} e o monitor de vídeo, a posição FF_{16} se comportando como registradores de 16 bits interagindo com a CPU como uma posição de memória qualquer.

Conjunto de instruções Existem 16 instruções, mas só vamos usar 14. Cada instrução tem 2 partes o *opcode* e o(s) operando(s). O opcode indica o que fazer com o operando. Assim, a instrução (de 16 bits) está dividida em 4 campos de 4 bits cada. O primeiro sub-campo é o opcode e os outros 12 são interpretados de acordo com o opcode. Veja em



Veja que nem todas as operações exigem três operandos. O que não é usado não precisa estar preenchido com zeros. Um endereço de registrador é indicado por um único dígito hexadecimal e usa um único sub-campo, enquanto um endereço de memória usa dois dígitos hexadecimais (portanto 2 sub-campos). Veja as instruções projetadas

Instrução	Código	Operandos			Ação
	d_1	d_2	d_3	d_4	
HALT	0				Para a execução do programa
LOAD	1	R_D	M_F		$R_D \leftarrow M_F$
STORE	2	M_D		R_F	$M_D \leftarrow R_F$
ADDI	3	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
ADDF	4	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
MOVE	5	R_D	R_F		$R_D \leftarrow R_F$
NOT	6	R_D	R_F		$R_D \leftarrow \overline{R}_F$
AND	7	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ AND R_{F_2}
OR	8	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ OR R_{F_2}
XOR	9	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ XOR R_{F_2}
INC	A	R			$R \leftarrow R + 1$
DEC	B	R			$R \leftarrow R - 1$
ROTATE	C	R	n	0 ou 1	Rot_n R
JUMP	D	R	n		Se $R_0 \neq$ R então PC = *n*, do co

Legenda: R_F, R_{F_1}, R_{F_2}: endereço hexadecimal dos registradores fonte
R_D: endereço hexadecimal do registrador destino
M_F: endereço hexadecimal da localização de memória fonte
M_D: endereço hexadecimal da localização da memória destino
n: número hexadecimal
d_1, d_2, d_3, d_4: primeiro, segundo, terceiro e quarto dígitos hexadecimais

Exemplo Acompanhe a execução da adição de dois inteiros A e B colocando o resultado em C. Em alto nível a instrução é $C = A + B$. Para isso, o computador precisa colocar os 2 operandos em registradores distintos (por exemplo R_0 e R_1) e determinar que o resultado vá para outro registrador (por exemplo R_2). A ULA só pode operar se os dados estiverem em registradores. Mas, como há um número limitado de registradores, há um constante fluxo de vai-e-vem entre eles e a memória. Por simplicidade assume-se que os dois valores estão em na posição 40_{16} e 41_{16} enquanto o resultado será colocado na posição 42. Desta maneira esta soma precisa 5 instruções:

1. Carrega o conteúdo de M_{40} em R_0
2. carrega o conteúdo de M_{41} em R_1
3. Adiciona R_0 com R_1 e coloca o resultado em R_2
4. Armazena R_2 em M_{42}
5. termina o programa

Veja o programa em linguagem de máquina

Código	Interpretação			
$(1040)_{16}$	1: LOAD	0: R_0	40: M_{40}	
$(1141)_{16}$	1: LOAD	1: R_1	41: M_{41}	
$(3201)_{16}$	3: ADDI	2: R_2	0: R_0	1: R_1
$(2422)_{16}$	2: STORE	42: M_{42}		2: R_2
$(0000)_{16}$	0: HALT			

Perceba que lá na máquina o programa é `1040114132012422 0000` (é mole ?).

Note que isto é só uma pequena parte da coisa. Antes, precisaríamos ter digitado um valor, que obviamente entraria na posição do teclado e transferido este valor para o endereço 40. Depois a mesma coisa indo para o endereço 41, depois a execução deste programa e finalmente a transferência do valor do endereço 42 para o monitor de vídeo. Ufa! Apenas para sua referência o programa completo é

$$\underbrace{1FFE}_{1}\ \underbrace{240F}_{2}\ \underbrace{1FFE}_{3}\ \underbrace{241F}_{4}\ \underbrace{1040}_{5}\ \underbrace{1141}_{6}\ \underbrace{3201}_{7}\ \underbrace{2422}_{8}$$

$$\underbrace{1F42}_{9}\ \underbrace{2FFF}_{10}\ \underbrace{0000}_{11}$$

As instruções 1 a 4 são para entrada e as 9 a 10 para saída.

Exemplo Nos exercicios a seguir, considere que os dados ja foram lidos nas posições de memória citadas e que serão exibidos a partir das posições também citadas. O que você deve resolver é a tradução das expressões aritméticas para linguagem de máquina e vice versa. Para facilitar a correção, use sempre o menor registrador disponível no momento.

Pede-se que você traduza para a linguagem de máquina a expressão $C \leftarrow A + (2 * B)$, sendo que &A=61, &B=65 e &D=72. Eis como ficaria

```
1065 - LOAD,0,65
3100 - ADDI,1,0,0  // 1=0+0
1261 - LOAD,2,61
3312 - ADDI,3,1,2  // 3=1+2
2723 - STORE,72,3
em linguagem de máquina 1065.3100.1261.3312.2723
```

Pede-se também o que faz o programa

```
1060.3100.3210.B200.B211.2732
```

A instrução `1060`, carrega no R_0 o conteúdo do endereço 60.
A instrução `3100`, soma os conteudos de R_0 e R_0 colocando o resultado em R_1, ou $R_1 = R_0 + R_0$
A instrução `3210`, soma os conteudos de R_1 e R_0 colocando o resultado em R_2, ou $R_2 = R_1 + R_0$ que devidamente interpretado com o anterior faz $R_2 = 3 \times R_0$
Depois, a instrução `B200`, decrementa o R_2. Note que o `00` que termina a instrução é puro lixo e não interessa. Outra maneira de representar esta instrução é `R2??` Mais um `B211`, que novamente decrementa o R_2. Agora o lixo é `11`
Finalmente, `2732` armazena o R_2 no endereço 73. Em resumo, aquele programa calcula $(X)_{73} \leftarrow ((Y)_{60} \times 3) - 2$

☞ Para você fazer

1. Traduza para linguagem de máquina a expressão $B = A + 3$,com &A=45,&B=60
2. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.A0??.2700
3. Traduza para linguagem de máquina a expressão $B = A * 2$,com &A=45,&B=70
4. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.3100.3101.2701

1

2

3

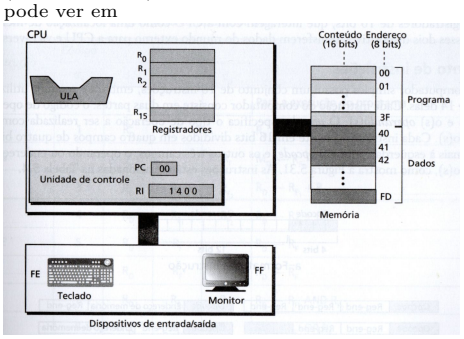
4

205-76506 - n lim

CEP - UP - UTFPR - PUCPr - UFPR –
10/06/2024 - 08:33:04.1
Prof Dr P Kantek (pkantek@gmail.com)
Um computador simples VIVXq06a, V: 1.31
76663 DANILO DEPETRIS SOARES 24CC2205 - 2 27/jun limite 1. avaliacao _____ / _____ /

Um computador simples

Vamos projetar um computador simples, para estudar de perto a sua arquitetura e seu processamento de instruções. O fato de ser um computador teórico (portanto irreal) não atrapalha. Projetado como se pode ver em



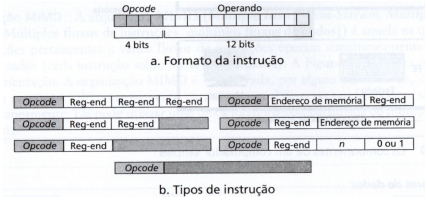
o computador terá 3 componentes: CPU, memória e subsistema de E/S. A CPU é dividida em 3 seções: registradores de dados, unidade aritmético-lógica e unidade de controle.

Existem 16 registradores de dados de 16 bits cada, e com endereços hexadecimais 0,1,...,F, conhecidos como $R_0, R_1, ..., R_{15}$. A unidade de controle controla as operações da ULA e os acessos à memória e ao subsistema de E/S. Tem 2 registradores dedicados: o PC (*Program Counter*) que aponta para o endereço de memória que contém a próxima instrução e o RI (*registrador de instrução*) mantém a instrução que está sendo executada no ciclo corrente. Após cada ciclo, o PC é incrementado (quase sempre) em 1 e o RI é atualizado com a próxima instrução.

A memória principal tem 256 localizações de 16 bits cada. Seus endereços binários variam entre 00000000_2 e 11111101_2 ou entre 00_{16} e FD_{16}. A memória contém dados e instruções. As primeiras 64 localizações (de 00 a $3F_{16}$ são dedicadas a instruções que são armazenadas em posições consecutivas. Já a área de 40 a FD_{16} são utilizadas para armazenar dados.

Já o subsistema de E/S é bastante primitivo, consistindo de um teclado e um vídeo. Assume-se que o teclado usa a posição de memória FE_{16} e o monitor de vídeo, a posição FF_{16} se comportando como registradores de 16 bits interagindo com a CPU como uma posição de memória qualquer.

Conjunto de instruções Existem 16 instruções, mas só vamos usar 14. Cada instrução tem 2 partes o *opcode* e o(s) operando(s). O opcode indica o que fazer com o operando. Assim, a instrução (de 16 bits) está dividida em 4 campos de 4 bits cada. O primeiro sub-campo é o opcode e os outros 12 são interpretados de acordo com o opcode. Veja em



Veja que nem todas as operações exigem três operandos. O que não é usado não precisa estar preenchido com zeros. Um endereço de registrador é indicado por um único dígito hexadecimal e usa um único sub-campo, enquanto um endereço de memória usa dois dígitos hexadecimais (portanto 2 sub-campos). Veja as instruções projetadas

Instrução	Código	Operandos			Ação
	d_1	d_2	d_3	d_4	
HALT	0				Para a execução do programa
LOAD	1	R_D	M_F		$R_D \leftarrow M_F$
STORE	2	M_D		R_F	$M_D \leftarrow R_F$
ADDI	3	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
ADDF	4	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
MOVE	5	R_D	R_F		$R_D \leftarrow R_F$
NOT	6	R_D	R_F		$R_D \leftarrow \overline{R}_F$
AND	7	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ AND R_{F_2}
OR	8	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ OR R_{F_2}
XOR	9	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ XOR R_{F_2}
INC	A	R			$R \leftarrow R + 1$
DEC	B	R			$R \leftarrow R - 1$
ROTATE	C	R	n	0 ou 1	Rot_n R
JUMP	D	R	n		Se $R_0 \neq$ R então PC = *n*, do co

Legenda: R_F, R_{F_1}, R_{F_2}: endereço hexadecimal dos registradores fonte
R_D: endereço hexadecimal do registrador destino
M_F: endereço hexadecimal da localização de memória fonte
M_D: endereço hexadecimal da localização da memória destino
n: número hexadecimal
d_1, d_2, d_3, d_4: primeiro, segundo, terceiro e quarto dígitos hexadecimais

Exemplo Acompanhe a execução da adição de dois inteiros A e B colocando o resultado em C. Em alto nível a instrução é $C = A + B$. Para isso, o computador precisa colocar os 2 operandos em registradores distintos (por exemplo R_0 e R_1) e determinar que o resultado vá para outro registrador (por exemplo R_2). A ULA só pode operar se os dados estiverem em registradores. Mas, como há um número limitado de registradores, há um constante fluxo de vai-e-vem entre eles e a memória. Por simplicidade assume-se que os dois valores estão em na posição 40_{16} e 41_{16} enquanto o resultado será colocado na posição 42. Desta maneira esta soma precisa 5 instruções:

1. Carrega o conteúdo de M_{40} em R_0
2. carrega o conteúdo de M_{41} em R_1
3. Adiciona R_0 com R_1 e coloca o resultado em R_2
4. Armazena R_2 em M_{42}
5. termina o programa

Veja o programa em linguagem de máquina

Código	Interpretação			
$(1040)_{16}$	1: LOAD	0: R_0	40: M_{40}	
$(1141)_{16}$	1: LOAD	1: R_1	41: M_{41}	
$(3201)_{16}$	3: ADDI	2: R_2	0: R_0	1: R_1
$(2422)_{16}$	2: STORE	42: M_{42}		2: R_2
$(0000)_{16}$	0: HALT			

Perceba que lá na máquina o programa é `10401141320124220000` (é mole ?).

Note que isto é só uma pequena parte da coisa. Antes, precisaríamos ter digitado um valor, que obviamente entraria na posição do teclado e transferido este valor para o endereço 40. Depois a mesma coisa indo para o endereço 41, depois a execução deste programa e finalmente a transferência do valor do endereço 42 para o monitor de vídeo. Ufa! Apenas para sua referência o programa completo é

$$\underbrace{1FFE}_{1}\underbrace{240F}_{2}\underbrace{1FFE}_{3}\underbrace{241F}_{4}\underbrace{1040}_{5}\underbrace{1141}_{6}\underbrace{3201}_{7}\underbrace{2422}_{8}$$

$$\underbrace{1F42}_{9}\underbrace{2FFF}_{10}\underbrace{0000}_{11}$$

As instruções 1 a 4 são para entrada e as 9 a 10 para saída.

Exemplo Nos exercicios a seguir, considere que os dados ja foram lidos nas posições de memória citadas e que serão exibidos a partir das posições também citadas. O que você deve resolver é a tradução das expressões aritméticas para linguagem de máquina e vice versa. Para facilitar a correção, use sempre o menor registrador disponível no momento.

Pede-se que você traduza para a linguagem de máquina a expressão $C \leftarrow A + (2 * B)$, sendo que &A=61, &B=65 e &D=72. Eis como ficaria

```
1065 - LOAD,0,65
3100 - ADDI,1,0,0  // 1=0+0
1261 - LOAD,2,61
3312 - ADDI,3,1,2  // 3=1+2
2723 - STORE,72,3
em linguagem de máquina 1065.3100.1261.3312.2723
```

Pede-se também o que faz o programa

```
1060.3100.3210.B200.B211.2732
```

A instrução `1060`, carrega no R_0 o conteúdo do endereço 60.
A instrução `3100`, soma os conteudos de R_0 e R_0 colocando o resultado em R_1, ou $R_1 = R_0 + R_0$
A instrução `3210`, soma os conteudos de R_1 e R_0 colocando o resultado em R_2, ou $R_2 = R_1 + R_0$ que devidamente interpretado com o anterior faz $R_2 = 3 \times R_0$
Depois, a instrução `B200`, decrementa o R_2. Note que o `00` que termina a instrução é puro lixo e não interessa. Outra maneira de representar esta instrução é `R2??` Mais um `B211`, que novamente decrementa o R_2. Agora o lixo é `11`
Finalmente, `2732` armazena o R_2 no endereço 73. Em resumo, aquele programa calcula $(X)_{73} \leftarrow ((Y)_{60} \times 3) - 2$

☞ Para você fazer

1. Traduza para linguagem de máquina a expressão $C = B - 1$,com &C=45,&B=60
2. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.3100.2701
3. Traduza para linguagem de máquina a expressão $B = A + 3$,com &A=45,&B=60
4. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.A0??.2700

1

2

3

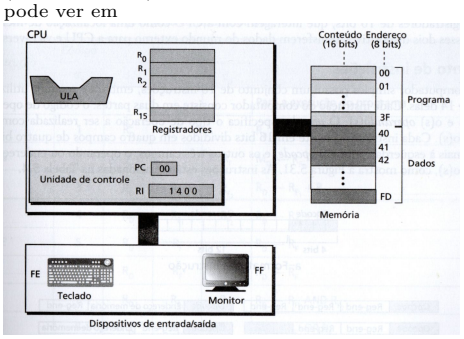
4

205-76663 - n lim

CEP - UP - UTFPR - PUCPr - UFPR –
10/06/2024 - 08:33:04.1
Prof Dr P Kantek (pkantek@gmail.com)
Um computador simples VIVXq06a, V: 1.31
76513 EDUARDO CAVALHEIRO DIAS
24CC2205 - 3 27/jun limite 1. avaliacao _____ /
/ _____

Um computador simples

Vamos projetar um computador simples, para estudar de perto a sua arquitetura e seu processamento de instruções. O fato de ser um computador teórico (portanto irreal) não atrapalha. Projetado como se pode ver em



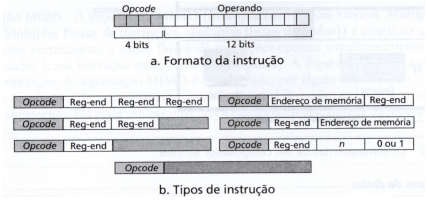
o computador terá 3 componentes: CPU, memória e subsistema de E/S. A CPU é dividida em 3 seções: registradores de dados, unidade aritmético-lógica e unidade de controle.

Existem 16 registradores de dados de 16 bits cada, e com endereços hexadecimais 0,1,...,F, conhecidos como $R_0, R_1, ..., R_{15}$. A unidade de controle controla as operações da ULA e os acessos à memória e ao subsistema de E/S. Tem 2 registradores dedicados: o PC (*Program Counter*) que aponta para o endereço de memória que contém a próxima instrução e o RI (*registrador de instrução*) mantém a instrução que está sendo executada no ciclo corrente. Após cada ciclo, o PC é incrementado (quase sempre) em 1 e o RI é atualizado com a próxima instrução.

A memória principal tem 256 localizações de 16 bits cada. Seus endereços binários variam entre 00000000_2 e 11111101_2 ou entre 00_{16} e FD_{16}. A memória contém dados e instruções. As primeiras 64 localizações (de 00 a $3F_{16}$ são dedicadas a instruções que são armazenadas em posições consecutivas. Já a área de 40 a FD_{16} são utilizadas para armazenar dados.

Já o subsistema de E/S é bastante primitivo, consistindo de um teclado e um vídeo. Assume-se que o teclado usa a posição de memória FE_{16} e o monitor de vídeo, a posição FF_{16} se comportando como registradores de 16 bits interagindo com a CPU como uma posição de memória qualquer.

Conjunto de instruções Existem 16 instruções, mas só vamos usar 14. Cada instrução tem 2 partes o *opcode* e o(s) operando(s). O opcode indica o que fazer com o operando. Assim, a instrução (de 16 bits) está dividida em 4 campos de 4 bits cada. O primeiro sub-campo é o opcode e os outros 12 são interpretados de acordo com o opcode. Veja em



Veja que nem todas as operações exigem três operandos. O que não é usado não precisa estar preenchido com zeros. Um endereço de registrador é indicado por um único dígito hexadecimal e usa um único sub-campo, enquanto um endereço de memória usa dois dígitos hexadecimais (portanto 2 sub-campos). Veja as instruções projetadas

Instrução	Código	Operandos			Ação
	d_1	d_2	d_3	d_4	
HALT	0				Para a execução do programa
LOAD	1	R_D	M_F		$R_D \leftarrow M_F$
STORE	2	M_D		R_F	$M_D \leftarrow R_F$
ADDI	3	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
ADDF	4	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
MOVE	5	R_D	R_F		$R_D \leftarrow R_F$
NOT	6	R_D	R_F		$R_D \leftarrow \overline{R}_F$
AND	7	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ AND R_{F_2}
OR	8	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ OR R_{F_2}
XOR	9	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ XOR R_{F_2}
INC	A	R			$R \leftarrow R + 1$
DEC	B	R			$R \leftarrow R - 1$
ROTATE	C	R	n	0 ou 1	Rot_n R
JUMP	D	R	n		Se $R_0 \neq$ R então PC = *n*, do co

Legenda: R_F, R_{F_1}, R_{F_2}: endereço hexadecimal dos registradores fonte
R_D: endereço hexadecimal do registrador destino
M_F: endereço hexadecimal da localização de memória fonte
M_D: endereço hexadecimal da localização da memória destino
n: número hexadecimal
d_1, d_2, d_3, d_4: primeiro, segundo, terceiro e quarto dígitos hexadecimais

Exemplo Acompanhe a execução da adição de dois inteiros A e B colocando o resultado em C. Em alto nível a instrução é $C = A + B$. Para isso, o computador precisa colocar os 2 operandos em registradores distintos (por exemplo R_0 e R_1) e determinar que o resultado vá para outro registrador (por exemplo R_2). A ULA só pode operar se os dados estiverem em registradores. Mas, como há um número limitado de registradores, há um constante fluxo de vai-e-vem entre eles e a memória. Por simplicidade assume-se que os dois valores estão em na posição 40_{16} e 41_{16} enquanto o resultado será colocado na posição 42. Desta maneira esta soma precisa 5 instruções:

1. Carrega o conteúdo de M_{40} em R_0
2. carrega o conteúdo de M_{41} em R_1
3. Adiciona R_0 com R_1 e coloca o resultado em R_2
4. Armazena R_2 em M_{42}
5. termina o programa

Veja o programa em linguagem de máquina

Código	Interpretação			
$(1040)_{16}$	1: LOAD	0: R_0	40: M_{40}	
$(1141)_{16}$	1: LOAD	1: R_1	41: M_{41}	
$(3201)_{16}$	3: ADDI	2: R_2	0: R_0	1: R_1
$(2422)_{16}$	2: STORE	42: M_{42}		2: R_2
$(0000)_{16}$	0: HALT			

Perceba que lá na máquina o programa é `1040114132012422 0000` (é mole ?).

Note que isto é só uma pequena parte da coisa. Antes, precisaríamos ter digitado um valor, que obviamente entraria na posição do teclado e transferido este valor para o endereço 40. Depois a mesma coisa indo para o endereço 41, depois a execução deste programa e finalmente a transferência do valor do endereço 42 para o monitor de vídeo. Ufa! Apenas para sua referência o programa completo é

$$\underbrace{1FFE}_{1}\underbrace{240F}_{2}\underbrace{1FFE}_{3}\underbrace{241F}_{4}\underbrace{1040}_{5}\underbrace{1141}_{6}\underbrace{3201}_{7}\underbrace{2422}_{8}$$

$$\underbrace{1F42}_{9}\underbrace{2FFF}_{10}\underbrace{0000}_{11}$$

As instruções 1 a 4 são para entrada e as 9 a 10 para saída.

Exemplo Nos exercicios a seguir, considere que os dados ja foram lidos nas posições de memória citadas e que serão exibidos a partir das posições também citadas. O que você deve resolver é a tradução das expressões aritméticas para linguagem de máquina e vice versa. Para facilitar a correção, use sempre o menor registrador disponível no momento.

Pede-se que você traduza para a linguagem de máquina a expressão $C \leftarrow A + (2 * B)$, sendo que &A=61, &B=65 e &D=72. Eis como ficaria

```
1065 - LOAD,0,65
3100 - ADDI,1,0,0  // 1=0+0
1261 - LOAD,2,61
3312 - ADDI,3,1,2  // 3=1+2
2723 - STORE,72,3
em linguagem de máquina 1065.3100.1261.3312.2723
```

Pede-se também o que faz o programa

```
1060.3100.3210.B200.B211.2732
```

A instrução `1060`, carrega no R_0 o conteúdo do endereço 60.
A instrução `3100`, soma os conteudos de R_0 e R_0 colocando o resultado em R_1, ou $R_1 = R_0 + R_0$
A instrução `3210`, soma os conteudos de R_1 e R_0 colocando o resultado em R_2, ou $R_2 = R_1 + R_0$ que devidamente interpretado com o anterior faz $R_2 = 3 \times R_0$
Depois, a instrução `B200`, decrementa o R_2. Note que o `00` que termina a instrução é puro lixo e não interessa. Outra maneira de representar esta instrução é `R2??` Mais um `B211`, que novamente decrementa o R_2. Agora o lixo é `11`
Finalmente, `2732` armazena o R_2 no endereço 73. Em resumo, aquele programa calcula $(X)_{73} \leftarrow ((Y)_{60} \times 3) - 2$

☞ Para você fazer

1. Traduza para linguagem de máquina a expressão $D = B + 1$,com &B=45,&D=70
2. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.3100.2701
3. Traduza para linguagem de máquina a expressão $B = A + 3$,com &A=45,&B=60
4. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1060.A0??.A0??.2450

1

2

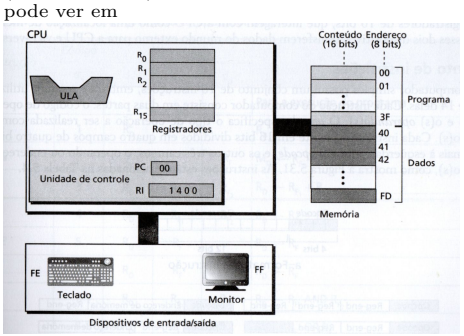
3

4

CEP - UP - UTFPR - PUCPr - UFPR –
10/06/2024 - 08:33:04.1
Prof Dr P Kantek (pkantek@gmail.com)
Um computador simples VIVXq06a, V: 1.31
76520 ERIC CORDEIRO 24CC2205 - 4 27/jun
limite 1. avaliacao _____ / _____ / _____

Um computador simples

Vamos projetar um computador simples, para estudar de perto a sua arquitetura e seu processamento de instruções. O fato de ser um computador teórico (portanto irreal) não atrapalha. Projetado como se pode ver em



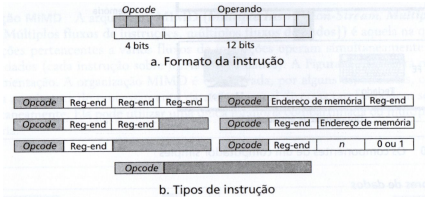
o computador terá 3 componentes: CPU, memória e subsistema de E/S. A CPU é dividida em 3 seções: registradores de dados, unidade aritmético-lógica e unidade de controle.

Existem 16 registradores de dados de 16 bits cada, e com endereços hexadecimais 0,1,...,F, conhecidos como $R_0, R_1, ..., R_{15}$. A unidade de controle controla as operações da ULA e os acessos à memória e ao subsistema de E/S. Tem 2 registradores dedicados: o PC (*Program Counter*) que aponta para o endereço de memória que contém a próxima instrução e o RI (*registrador de instrução*) mantém a instrução que está sendo executada no ciclo corrente. Após cada ciclo, o PC é incrementado (quase sempre) em 1 e o RI é atualizado com a próxima instrução.

A memória principal tem 256 localizações de 16 bits cada. Seus endereços binários variam entre 00000000_2 e 11111101_2 ou entre 00_{16} e FD_{16}. A memória contém dados e instruções. As primeiras 64 localizações (de 00 a $3F_{16}$ são dedicadas a instruções que são armazenadas em posições consecutivas. Já a área de 40 a FD_{16} são utilizadas para armazenar dados.

Já o subsistema de E/S é bastante primitivo, consistindo de um teclado e um vídeo. Assume-se que o teclado usa a posição de memória FE_{16} e o monitor de vídeo, a posição FF_{16} se comportando como registradores de 16 bits interagindo com a CPU como uma posição de memória qualquer.

Conjunto de instruções Existem 16 instruções, mas só vamos usar 14. Cada instrução tem 2 partes o *opcode* e o(s) operando(s). O opcode indica o que fazer com o operando. Assim, a instrução (de 16 bits) está dividida em 4 campos de 4 bits cada. O primeiro sub-campo é o opcode e os outros 12 são interpretados de acordo com o opcode. Veja em



Veja que nem todas as operações exigem três operandos. O que não é usado não precisa estar preenchido com zeros. Um endereço de registrador é indicado por um único dígito hexadecimal e usa um único sub-campo, enquanto um endereço de memória usa dois dígitos hexadecimais (portanto 2 sub-campos). Veja as instruções projetadas

Instrução	Código	Operandos				Ação
	d_1	d_2	d_3	d_4		
HALT	0					Para a execução do programa
LOAD	1	R_D	M_F			$R_D \leftarrow M_F$
STORE	2	M_D		R_F		$M_D \leftarrow R_F$
ADDI	3	R_D	R_{F_1}	R_{F_2}		$R_D \leftarrow R_{F_1} + R_{F_2}$
ADDF	4	R_D	R_{F_1}	R_{F_2}		$R_D \leftarrow R_{F_1} + R_{F_2}$
MOVE	5	R_D	R_F			$R_D \leftarrow R_F$
NOT	6	R_D	R_F			$R_D \leftarrow \overline{R}_F$
AND	7	R_D	R_{F_1}	R_{F_2}		$R_D \leftarrow R_{F_1}$ AND R_{F_2}
OR	8	R_D	R_{F_1}	R_{F_2}		$R_D \leftarrow R_{F_1}$ OR R_{F_2}
XOR	9	R_D	R_{F_1}	R_{F_2}		$R_D \leftarrow R_{F_1}$ XOR R_{F_2}
INC	A	R				$R \leftarrow R + 1$
DEC	B	R				$R \leftarrow R - 1$
ROTATE	C	R	n	0 ou 1		Rot_n R
JUMP	D	R	n			Se $R_0 \neq$ R então PC = *n*, do co

Legenda: R_F, R_{F_1}, R_{F_2}: endereço hexadecimal dos registradores fonte
R_D: endereço hexadecimal do registrador destino
M_F: endereço hexadecimal da localização de memória fonte
M_D: endereço hexadecimal da localização da memória destino
n: número hexadecimal
d_1, d_2, d_3, d_4: primeiro, segundo, terceiro e quarto dígitos hexadecimais

Exemplo Acompanhe a execução da adição de dois inteiros A e B colocando o resultado em C. Em alto nível a instrução é $C = A + B$. Para isso, o computador precisa colocar os 2 operandos em registradores distintos (por exemplo R_0 e R_1) e determinar que o resultado vá para outro registrador (por exemplo R_2). A ULA só pode operar se os dados estiverem em registradores. Mas, como há um número limitado de registradores, há um constante fluxo de vai-e-vem entre eles e a memória. Por simplicidade assume-se que os dois valores estão em na posição 40_{16} e 41_{16} enquanto o resultado será colocado na posição 42. Desta maneira esta soma precisa 5 instruções:

1. Carrega o conteúdo de M_{40} em R_0
2. carrega o conteúdo de M_{41} em R_1
3. Adiciona R_0 com R_1 e coloca o resultado em R_2
4. Armazena R_2 em M_{42}
5. termina o programa

Veja o programa em linguagem de máquina

Código	Interpretação			
$(1040)_{16}$	1: LOAD	0: R_0	40: M_{40}	
$(1141)_{16}$	1: LOAD	1: R_1	41: M_{41}	
$(3201)_{16}$	3: ADDI	2: R_2	0: R_0	1: R_1
$(2422)_{16}$	2: STORE	42: M_{42}		2: R_2
$(0000)_{16}$	0: HALT			

Perceba que lá na máquina o programa é `1040114132012422000`0 (é mole ?).

Note que isto é só uma pequena parte da coisa. Antes, precisaríamos ter digitado um valor, que obviamente entraria na posição do teclado e transferido este valor para o endereço 40. Depois a mesma coisa indo para o endereço 41, depois a execução deste programa e finalmente a transferência do valor do endereço 42 para o monitor de vídeo. Ufa! Apenas para sua referência o programa completo é

$$\underbrace{1FFE}_{1}\ \underbrace{240F}_{2}\ \underbrace{1FFE}_{3}\ \underbrace{241F}_{4}\ \underbrace{1040}_{5}\ \underbrace{1141}_{6}\ \underbrace{3201}_{7}\ \underbrace{2422}_{8}$$

$$\underbrace{1F42}_{9}\ \underbrace{2FFF}_{10}\ \underbrace{0000}_{11}$$

As instruções 1 a 4 são para entrada e as 9 a 10 para saída.

Exemplo Nos exercicios a seguir, considere que os dados ja foram lidos nas posições de memória citadas e que serão exibidos a partir das posições também citadas. O que você deve resolver é a tradução das expressões aritméticas para linguagem de máquina e vice versa. Para facilitar a correção, use sempre o menor registrador disponível no momento.

Pede-se que você traduza para a linguagem de máquina a expressão $C \leftarrow A + (2 * B)$, sendo que &A=61, &B=65 e &D=72. Eis como ficaria

```
1065 - LOAD,0,65
3100 - ADDI,1,0,0  // 1=0+0
1261 - LOAD,2,61
3312 - ADDI,3,1,2  // 3=1+2
2723 - STORE,72,3
em linguagem de máquina 1065.3100.1261.3312.2723
```

Pede-se também o que faz o programa

`1060.3100.3210.B200.B211.2732`

A instrução `1060`, carrega no R_0 o conteúdo do endereço 60.
A instrução `3100`, soma os conteudos de R_0 e R_0 colocando o resultado em R_1, ou $R_1 = R_0 + R_0$
A instrução `3210`, soma os conteudos de R_1 e R_0 colocando o resultado em R_2, ou $R_2 = R_1 + R_0$ que devidamente interpretado com o anterior faz $R_2 = 3 \times R_0$
Depois, a instrução `B200`, decrementa o R_2. Note que o `00` que termina a instrução é puro lixo e não interessa. Outra maneira de representar esta instrução é `R2??` Mais um `B211`, que novamente decrementa o R_2. Agora o lixo é `11`
Finalmente, `2732` armazena o R_2 no endereço 73. Em resumo, aquele programa calcula $(X)_{73} \leftarrow ((Y)_{60} \times 3) - 2$

☞ Para você fazer

1. Traduza para linguagem de máquina a expressão $D = B + 1$,com &B=45,&D=70
2. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.3100.3101.2701
3. Traduza para linguagem de máquina a expressão $C = B - 1$,com &C=45,&B=60
4. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.3100.2701

1

2

3

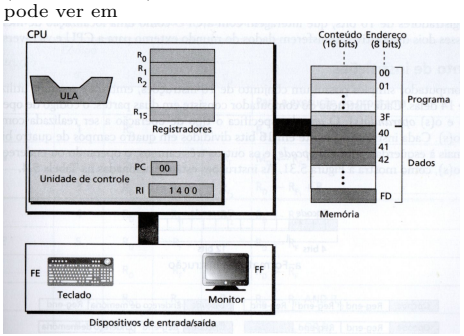
4

205-76520 - n lim

CEP - UP - UTFPR - PUCPr - UFPR –
10/06/2024 - 08:33:04.1
Prof Dr P Kantek (pkantek@gmail.com)
Um computador simples VIVXq06a, V: 1.31
76537 FELIPE EIJI KANO 24CC2205 - 5 27/jun
limite 1. avaliacao _____ / _____ / _____

Um computador simples

Vamos projetar um computador simples, para estudar de perto a sua arquitetura e seu processamento de instruções. O fato de ser um computador teórico (portanto irreal) não atrapalha. Projetado como se pode ver em



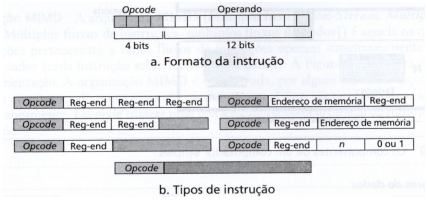
o computador terá 3 componentes: CPU, memória e subsistema de E/S. A CPU é dividida em 3 seções: registradores de dados, unidade aritmético-lógica e unidade de controle.

Existem 16 registradores de dados de 16 bits cada, e com endereços hexadecimais 0,1,...,F, conhecidos como $R_0, R_1, ..., R_{15}$. A unidade de controle controla as operações da ULA e os acessos à memória e ao subsistema de E/S. Tem 2 registradores dedicados: o PC (*Program Counter*) que aponta para o endereço de memória que contém a próxima instrução e o RI (*registrador de instrução*) mantém a instrução que está sendo executada no ciclo corrente. Após cada ciclo, o PC é incrementado (quase sempre) em 1 e o RI é atualizado com a próxima instrução.

A memória principal tem 256 localizações de 16 bits cada. Seus endereços binários variam entre 00000000_2 e 11111101_2 ou entre 00_{16} e FD_{16}. A memória contém dados e instruções. As primeiras 64 localizações (de 00 a $3F_{16}$ são dedicadas a instruções que são armazenadas em posições consecutivas. Já a área de 40 a FD_{16} são utilizadas para armazenar dados.

Já o subsistema de E/S é bastante primitivo, consistindo de um teclado e um vídeo. Assume-se que o teclado usa a posição de memória FE_{16} e o monitor de vídeo, a posição FF_{16} se comportando como registradores de 16 bits interagindo com a CPU como uma posição de memória qualquer.

Conjunto de instruções Existem 16 instruções, mas só vamos usar 14. Cada instrução tem 2 partes o *opcode* e o(s) operando(s). O opcode indica o que fazer com o operando. Assim, a instrução (de 16 bits) está dividida em 4 campos de 4 bits cada. O primeiro sub-campo é o opcode e os outros 12 são interpretados de acordo com o opcode. Veja em



Veja que nem todas as operações exigem três operandos. O que não é usado não precisa estar preenchido com zeros. Um endereço de registrador é indicado por um único dígito hexadecimal e usa um único sub-campo, enquanto um endereço de memória usa dois dígitos hexadecimais (portanto 2 sub-campos). Veja as instruções projetadas

Instrução	Código	Operandos			Ação
	d_1	d_2	d_3	d_4	
HALT	0				Para a execução do programa
LOAD	1	R_D	M_F		$R_D \leftarrow M_F$
STORE	2	M_D		R_F	$M_D \leftarrow R_F$
ADDI	3	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
ADDF	4	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
MOVE	5	R_D	R_F		$R_D \leftarrow R_F$
NOT	6	R_D	R_F		$R_D \leftarrow \overline{R}_F$
AND	7	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ AND R_{F_2}
OR	8	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ OR R_{F_2}
XOR	9	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ XOR R_{F_2}
INC	A	R			$R \leftarrow R + 1$
DEC	B	R			$R \leftarrow R - 1$
ROTATE	C	R	n	0 ou 1	Rot_n R
JUMP	D	R	n		Se $R_0 \neq$ R então PC = *n*, do co

Legenda: R_F, R_{F_1}, R_{F_2}: endereço hexadecimal dos registradores fonte
R_D: endereço hexadecimal do registrador destino
M_F: endereço hexadecimal da localização de memória fonte
M_D: endereço hexadecimal da localização da memória destino
n: número hexadecimal
d_1, d_2, d_3, d_4: primeiro, segundo, terceiro e quarto dígitos hexadecimais

Exemplo Acompanhe a execução da adição de dois inteiros A e B colocando o resultado em C. Em alto nível a instrução é $C = A + B$. Para isso, o computador precisa colocar os 2 operandos em registradores distintos (por exemplo R_0 e R_1) e determinar que o resultado vá para outro registrador (por exemplo R_2). A ULA só pode operar se os dados estiverem em registradores. Mas, como há um número limitado de registradores, há um constante fluxo de vai-e-vem entre eles e a memória. Por simplicidade assume-se que os dois valores estão em na posição 40_{16} e 41_{16} enquanto o resultado será colocado na posição 42. Desta maneira esta soma precisa 5 instruções:

1. Carrega o conteúdo de M_{40} em R_0
2. carrega o conteúdo de M_{41} em R_1
3. Adiciona R_0 com R_1 e coloca o resultado em R_2
4. Armazena R_2 em M_{42}
5. termina o programa

Veja o programa em linguagem de máquina

Código	Interpretação			
$(1040)_{16}$	1: LOAD	0: R_0	40: M_{40}	
$(1141)_{16}$	1: LOAD	1: R_1	41: M_{41}	
$(3201)_{16}$	3: ADDI	2: R_2	0: R_0	1: R_1
$(2422)_{16}$	2: STORE	42: M_{42}		2: R_2
$(0000)_{16}$	0: HALT			

Perceba que lá na máquina o programa é `1040114132012422 0000` (é mole ?).

Note que isto é só uma pequena parte da coisa. Antes, precisaríamos ter digitado um valor, que obviamente entraria na posição do teclado e transferido este valor para o endereço 40. Depois a mesma coisa indo para o endereço 41, depois a execução deste programa e finalmente a transferência do valor do endereço 42 para o monitor de vídeo. Ufa! Apenas para sua referência o programa completo é

$$\underbrace{1FFE}_{1}\ \underbrace{240F}_{2}\ \underbrace{1FFE}_{3}\ \underbrace{241F}_{4}\ \underbrace{1040}_{5}\ \underbrace{1141}_{6}\ \underbrace{3201}_{7}\ \underbrace{2422}_{8}$$

$$\underbrace{1F42}_{9}\ \underbrace{2FFF}_{10}\ \underbrace{0000}_{11}$$

As instruções 1 a 4 são para entrada e as 9 a 10 para saída.

Exemplo Nos exercicios a seguir, considere que os dados ja foram lidos nas posições de memória citadas e que serão exibidos a partir das posições também citadas. O que você deve resolver é a tradução das expressões aritméticas para linguagem de máquina e vice versa. Para facilitar a correção, use sempre o menor registrador disponível no momento.

Pede-se que você traduza para a linguagem de máquina a expressão $C \leftarrow A + (2 * B)$, sendo que &A=61, &B=65 e &D=72. Eis como ficaria

```
1065 - LOAD,0,65
3100 - ADDI,1,0,0  // 1=0+0
1261 - LOAD,2,61
3312 - ADDI,3,1,2  // 3=1+2
2723 - STORE,72,3
em linguagem de máquina 1065.3100.1261.3312.2723
```

Pede-se também o que faz o programa

```
1060.3100.3210.B200.B211.2732
```

A instrução `1060`, carrega no R_0 o conteúdo do endereço 60.
A instrução `3100`, soma os conteudos de R_0 e R_0 colocando o resultado em R_1, ou $R_1 = R_0 + R_0$
A instrução `3210`, soma os conteudos de R_1 e R_0 colocando o resultado em R_2, ou $R_2 = R_1 + R_0$ que devidamente interpretado com o anterior faz $R_2 = 3 \times R_0$
Depois, a instrução `B200`, decrementa o R_2. Note que o `00` que termina a instrução é puro lixo e não interessa. Outra maneira de representar esta instrução é `R2??` Mais um `B211`, que novamente decrementa o R_2. Agora o lixo é `11`
Finalmente, `2732` armazena o R_2 no endereço 73. Em resumo, aquele programa calcula $(X)_{73} \leftarrow ((Y)_{60} \times 3) - 2$

☞ Para você fazer

1. Traduza para linguagem de máquina a expressão $D = A + B + C$,com &A=40,&B=50,&C=60,&D=70
2. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.3100.3101.2701
3. Traduza para linguagem de máquina a expressão $D = A + A + A + A + A$, &A=45,&D=60
4. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.B0??.B0??.2600

1

2

3

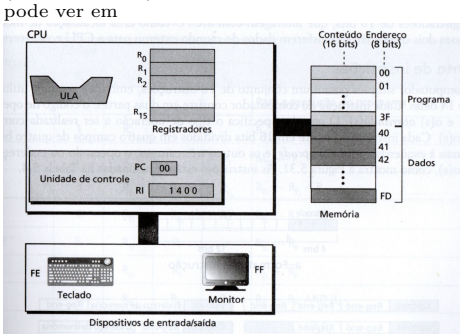
4

205-76537 - n lim

CEP - UP - UTFPR - PUCPr - UFPR –
10/06/2024 - 08:33:04.1
Prof Dr P Kantek (pkantek@gmail.com)
Um computador simples VIVXq06a, V: 1.31 76544
FELIPE JESMIEL LEITE 24CC2205 - 6 27/jun
limite 1. avaliacao _____ / _____ / _____

Um computador simples

Vamos projetar um computador simples, para estudar de perto a sua arquitetura e seu processamento de instruções. O fato de ser um computador teórico (portanto irreal) não atrapalha. Projetado como se pode ver em



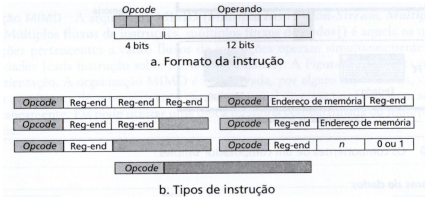
o computador terá 3 componentes: CPU, memória e subsistema de E/S. A CPU é dividida em 3 seções: registradores de dados, unidade aritmético-lógica e unidade de controle.

Existem 16 registradores de dados de 16 bits cada, e com endereços hexadecimais 0,1,...,F, conhecidos como $R_0, R_1, ..., R_{15}$. A unidade de controle controla as operações da ULA e os acessos à memória e ao subsistema de E/S. Tem 2 registradores dedicados: o PC (*Program Counter*) que aponta para o endereço de memória que contém a próxima instrução e o RI (*registrador de instrução*) mantém a instrução que está sendo executada no ciclo corrente. Após cada ciclo, o PC é incrementado (quase sempre) em 1 e o RI é atualizado com a próxima instrução.

A memória principal tem 256 localizações de 16 bits cada. Seus endereços binários variam entre 00000000_2 e 11111101_2 ou entre 00_{16} e FD_{16}. A memória contém dados e instruções. As primeiras 64 localizações (de 00 a $3F_{16}$ são dedicadas a instruções que são armazenadas em posições consecutivas. Já a área de 40 a FD_{16} são utilizadas para armazenar dados.

Já o subsistema de E/S é bastante primitivo, consistindo de um teclado e um vídeo. Assume-se que o teclado usa a posição de memória FE_{16} e o monitor de vídeo, a posição FF_{16} se comportando como registradores de 16 bits interagindo com a CPU como uma posição de memória qualquer.

Conjunto de instruções Existem 16 instruções, mas só vamos usar 14. Cada instrução tem 2 partes o *opcode* e o(s) operando(s). O opcode indica o que fazer com o operando. Assim, a instrução (de 16 bits) está dividida em 4 campos de 4 bits cada. O primeiro sub-campo é o opcode e os outros 12 são interpretados de acordo com o opcode. Veja em



Veja que nem todas as operações exigem três operandos. O que não é usado não precisa estar preenchido com zeros. Um endereço de registrador é indicado por um único dígito hexadecimal e usa um único sub-campo, enquanto um endereço de memória usa dois dígitos hexadecimais (portanto 2 sub-campos). Veja as instruções projetadas

Instrução	Código	Operandos			Ação
	d_1	d_2	d_3	d_4	
HALT	0				Para a execução do programa
LOAD	1	R_D	M_F		$R_D \leftarrow M_F$
STORE	2	M_D		R_F	$M_D \leftarrow R_F$
ADDI	3	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
ADDF	4	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
MOVE	5	R_D	R_F		$R_D \leftarrow R_F$
NOT	6	R_D	R_F		$R_D \leftarrow \overline{R}_F$
AND	7	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ AND R_{F_2}
OR	8	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ OR R_{F_2}
XOR	9	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ XOR R_{F_2}
INC	A	R			$R \leftarrow R + 1$
DEC	B	R			$R \leftarrow R - 1$
ROTATE	C	R	n	0 ou 1	Rot_n R
JUMP	D	R	n		Se $R_0 \neq$ R então PC = *n*, do co

Legenda: R_F, R_{F_1}, R_{F_2}: endereço hexadecimal dos registradores fonte
R_D: endereço hexadecimal do registrador destino
M_F: endereço hexadecimal da localização de memória fonte
M_D: endereço hexadecimal da localização da memória destino
n: número hexadecimal
d_1, d_2, d_3, d_4: primeiro, segundo, terceiro e quarto dígitos hexadecimais

Exemplo Acompanhe a execução da adição de dois inteiros A e B colocando o resultado em C. Em alto nível a instrução é $C = A + B$. Para isso, o computador precisa colocar os 2 operandos em registradores distintos (por exemplo R_0 e R_1) e determinar que o resultado vá para outro registrador (por exemplo R_2). A ULA só pode operar se os dados estiverem em registradores. Mas, como há um número limitado de registradores, há um constante fluxo de vai-e-vem entre eles e a memória. Por simplicidade assume-se que os dois valores estão em na posição 40_{16} e 41_{16} enquanto o resultado será colocado na posição 42. Desta maneira esta soma precisa 5 instruções:

1. Carrega o conteúdo de M_{40} em R_0
2. carrega o conteúdo de M_{41} em R_1
3. Adiciona R_0 com R_1 e coloca o resultado em R_2
4. Armazena R_2 em M_{42}
5. termina o programa

Veja o programa em linguagem de máquina

Código	Interpretação			
$(1040)_{16}$	1: LOAD	0: R_0	40: M_{40}	
$(1141)_{16}$	1: LOAD	1: R_1	41: M_{41}	
$(3201)_{16}$	3: ADDI	2: R_2	0: R_0	1: R_1
$(2422)_{16}$	2: STORE	42: M_{42}		2: R_2
$(0000)_{16}$	0: HALT			

Perceba que lá na máquina o programa é `1040114132012422 0000` (é mole ?).

Note que isto é só uma pequena parte da coisa. Antes, precisaríamos ter digitado um valor, que obviamente entraria na posição do teclado e transferido este valor para o endereço 40. Depois a mesma coisa indo para o endereço 41, depois a execução deste programa e finalmente a transferência do valor do endereço 42 para o monitor de vídeo. Ufa! Apenas para sua referência o programa completo é

$$\underbrace{1FFE}_{1}\underbrace{240F}_{2}\underbrace{1FFE}_{3}\underbrace{241F}_{4}\underbrace{1040}_{5}\underbrace{1141}_{6}\underbrace{3201}_{7}\underbrace{2422}_{8}$$

$$\underbrace{1F42}_{9}\underbrace{2FFF}_{10}\underbrace{0000}_{11}$$

As instruções 1 a 4 são para entrada e as 9 a 10 para saída.

Exemplo Nos exercicios a seguir, considere que os dados ja foram lidos nas posições de memória citadas e que serão exibidos a partir das posições também citadas. O que você deve resolver é a tradução das expressões aritméticas para linguagem de máquina e vice versa. Para facilitar a correção, use sempre o menor registrador disponível no momento.

Pede-se que você traduza para a linguagem de máquina a expressão $C \leftarrow A + (2 * B)$, sendo que &A=61, &B=65 e &D=72. Eis como ficaria

```
1065 - LOAD,0,65
3100 - ADDI,1,0,0  // 1=0+0
1261 - LOAD,2,61
3312 - ADDI,3,1,2  // 3=1+2
2723 - STORE,72,3
em linguagem de máquina 1065.3100.1261.3312.2723
```

Pede-se também o que faz o programa

`1060.3100.3210.B200.B211.2732`

A instrução `1060`, carrega no R_0 o conteúdo do endereço 60.
A instrução `3100`, soma os conteudos de R_0 e R_0 colocando o resultado em R_1, ou $R_1 = R_0 + R_0$
A instrução `3210`, soma os conteudos de R_1 e R_0 colocando o resultado em R_2, ou $R_2 = R_1 + R_0$ que devidamente interpretado com o anterior faz $R_2 = 3 \times R_0$
Depois, a instrução `B200`, decrementa o R_2. Note que o `00` que termina a instrução é puro lixo e não interessa. Outra maneira de representar esta instrução é `R2??` Mais um `B211`, que novamente decrementa o R_2. Agora o lixo é `11`
Finalmente, `2732` armazena o R_2 no endereço 73. Em resumo, aquele programa calcula $(X)_{73} \leftarrow ((Y)_{60} \times 3) - 2$

☞ Para você fazer

1. Traduza para linguagem de máquina a expressão $B = A - 2$,com &A=45,&B=60
2. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1060.A0??.A0??.2450
3. Traduza para linguagem de máquina a expressão $D = B + 1$,com &B=45,&D=70
4. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.3100.2701

1

2

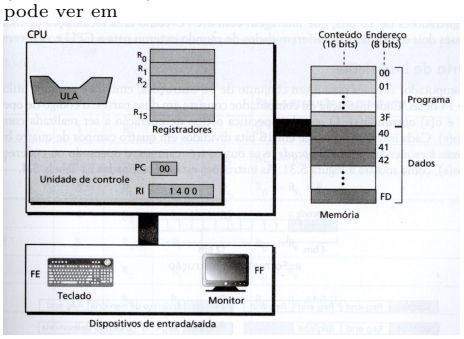
3

4

CEP - UP - UTFPR - PUCPr - UFPR –
10/06/2024 - 08:33:04.1
Prof Dr P Kantek (pkantek@gmail.com)
Um computador simples VIVXq06a, V: 1.31
76551 GABRIELA BERNARDO
PASQUALOTTO 24CC2205 - 7 27/jun limite 1.
avaliacao _____ / _____ / _____

Um computador simples

Vamos projetar um computador simples, para estudar de perto a sua arquitetura e seu processamento de instruções. O fato de ser um computador teórico (portanto irreal) não atrapalha. Projetado como se pode ver em



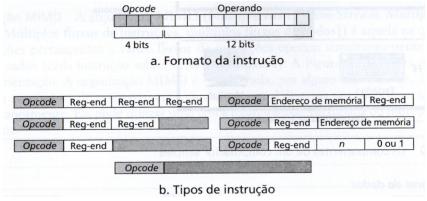
o computador terá 3 componentes: CPU, memória e subsistema de E/S. A CPU é dividida em 3 seções: registradores de dados, unidade aritmético-lógica e unidade de controle.

Existem 16 registradores de dados de 16 bits cada, e com endereços hexadecimais 0,1,...,F, conhecidos como $R_0, R_1, ..., R_{15}$. A unidade de controle controla as operações da ULA e os acessos à memória e ao subsistema de E/S. Tem 2 registradores dedicados: o PC (*Program Counter*) que aponta para o endereço de memória que contém a próxima instrução e o RI (*registrador de instrução*) mantém a instrução que está sendo executada no ciclo corrente. Após cada ciclo, o PC é incrementado (quase sempre) em 1 e o RI é atualizado com a próxima instrução.

A memória principal tem 256 localizações de 16 bits cada. Seus endereços binários variam entre 00000000_2 e 11111101_2 ou entre 00_{16} e FD_{16}. A memória contém dados e instruções. As primeiras 64 localizações (de 00 a $3F_{16}$ são dedicadas a instruções que são armazenadas em posições consecutivas. Já a área de 40 a FD_{16} são utilizadas para armazenar dados.

Já o subsistema de E/S é bastante primitivo, consistindo de um teclado e um vídeo. Assume-se que o teclado usa a posição de memória FE_{16} e o monitor de vídeo, a posição FF_{16} se comportando como registradores de 16 bits interagindo com a CPU como uma posição de memória qualquer.

Conjunto de instruções Existem 16 instruções, mas só vamos usar 14. Cada instrução tem 2 partes o *opcode* e o(s) operando(s). O opcode indica o que fazer com o operando. Assim, a instrução (de 16 bits) está dividida em 4 campos de 4 bits cada. O primeiro sub-campo é o opcode e os outros 12 são interpretados de acordo com o opcode. Veja em



Veja que nem todas as operações exigem três operandos. O que não é usado não precisa estar preenchido com zeros. Um endereço de registrador é indicado por um único dígito hexadecimal e usa um único sub-campo, enquanto um endereço de memória usa dois dígitos hexadecimais (portanto 2 sub-campos). Veja as instruções projetadas

Instrução	Código	Operandos			Ação
	d_1	d_2	d_3	d_4	
HALT	0				Para a execução do programa
LOAD	1	R_D	M_F		$R_D \leftarrow M_F$
STORE	2	M_D		R_F	$M_D \leftarrow R_F$
ADDI	3	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
ADDF	4	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
MOVE	5	R_D	R_F		$R_D \leftarrow R_F$
NOT	6	R_D	R_F		$R_D \leftarrow \overline{R}_F$
AND	7	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ AND R_{F_2}
OR	8	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ OR R_{F_2}
XOR	9	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ XOR R_{F_2}
INC	A	R			$R \leftarrow R + 1$
DEC	B	R			$R \leftarrow R - 1$
ROTATE	C	R	n	0 ou 1	Rot_n R
JUMP	D	R	n		Se $R_0 \neq$ R então PC = *n*, do co

Legenda: R_F, R_{F_1}, R_{F_2}: endereço hexadecimal dos registradores fonte
R_D: endereço hexadecimal do registrador destino
M_F: endereço hexadecimal da localização de memória fonte
M_D: endereço hexadecimal da localização da memória destino
n: número hexadecimal
d_1, d_2, d_3, d_4: primeiro, segundo, terceiro e quarto dígitos hexadecimais

Exemplo Acompanhe a execução da adição de dois inteiros A e B colocando o resultado em C. Em alto nível a instrução é $C = A + B$. Para isso, o computador precisa colocar os 2 operandos em registradores distintos (por exemplo R_0 e R_1) e determinar que o resultado vá para outro registrador (por exemplo R_2). A ULA só pode operar se os dados estiverem em registradores. Mas, como há um número limitado de registradores, há um constante fluxo de vai-e-vem entre eles e a memória. Por simplicidade assume-se que os dois valores estão em na posição 40_{16} e 41_{16} enquanto o resultado será colocado na posição 42. Desta maneira esta soma precisa 5 instruções:

1. Carrega o conteúdo de M_{40} em R_0
2. carrega o conteúdo de M_{41} em R_1
3. Adiciona R_0 com R_1 e coloca o resultado em R_2
4. Armazena R_2 em M_{42}
5. termina o programa

Veja o programa em linguagem de máquina

Código	Interpretação			
$(1040)_{16}$	1: LOAD	0: R_0	40: M_{40}	
$(1141)_{16}$	1: LOAD	1: R_1	41: M_{41}	
$(3201)_{16}$	3: ADDI	2: R_2	0: R_0	1: R_1
$(2422)_{16}$	2: STORE	42: M_{42}		2: R_2
$(0000)_{16}$	0: HALT			

Perceba que lá na máquina o programa é `1040114132012422 0000` (é mole ?).

Note que isto é só uma pequena parte da coisa. Antes, precisaríamos ter digitado um valor, que obviamente entraria na posição do teclado e transferido este valor para o endereço 40. Depois a mesma coisa indo para o endereço 41, depois a execução deste programa e finalmente a transferência do valor do endereço 42 para o monitor de vídeo. Ufa! Apenas para sua referência o programa completo é

$$\underbrace{1FFE}_{1}\ \underbrace{240F}_{2}\ \underbrace{1FFE}_{3}\ \underbrace{241F}_{4}\ \underbrace{1040}_{5}\ \underbrace{1141}_{6}\ \underbrace{3201}_{7}\ \underbrace{2422}_{8}$$

$$\underbrace{1F42}_{9}\ \underbrace{2FFF}_{10}\ \underbrace{0000}_{11}$$

As instruções 1 a 4 são para entrada e as 9 a 10 para saída.

Exemplo Nos exercicios a seguir, considere que os dados ja foram lidos nas posições de memória citadas e que serão exibidos a partir das posições também citadas. O que você deve resolver é a tradução das expressões aritméticas para linguagem de máquina e vice versa. Para facilitar a correção, use sempre o menor registrador disponível no momento.

Pede-se que você traduza para a linguagem de máquina a expressão $C \leftarrow A + (2 * B)$, sendo que &A=61, &B=65 e &D=72. Eis como ficaria

```
1065 - LOAD,0,65
3100 - ADDI,1,0,0  // 1=0+0
1261 - LOAD,2,61
3312 - ADDI,3,1,2  // 3=1+2
2723 - STORE,72,3
em linguagem de máquina 1065.3100.1261.3312.2723
```

Pede-se também o que faz o programa

```
1060.3100.3210.B200.B211.2732
```

A instrução `1060`, carrega no R_0 o conteúdo do endereço 60.
A instrução `3100`, soma os conteudos de R_0 e R_0 colocando o resultado em R_1, ou $R_1 = R_0 + R_0$
A instrução `3210`, soma os conteudos de R_1 e R_0 colocando o resultado em R_2, ou $R_2 = R_1 + R_0$ que devidamente interpretado com o anterior faz $R_2 = 3 \times R_0$
Depois, a instrução `B200`, decrementa o R_2. Note que o `00` que termina a instrução é puro lixo e não interessa. Outra maneira de representar esta instrução é `R2??` Mais um `B211`, que novamente decrementa o R_2. Agora o lixo é `11`
Finalmente, `2732` armazena o R_2 no endereço 73. Em resumo, aquele programa calcula $(X)_{73} \leftarrow ((Y)_{60} \times 3) - 2$

☞ Para você fazer

1. Traduza para linguagem de máquina a expressão $D = A + B + C$,com &A=40,&B=50,&C=60,&D=70
2. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.B0??.B0??.2600
3. Traduza para linguagem de máquina a expressão $D = A + 1$,com &A=45,&D=70
4. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.3100.2701

1

2

3

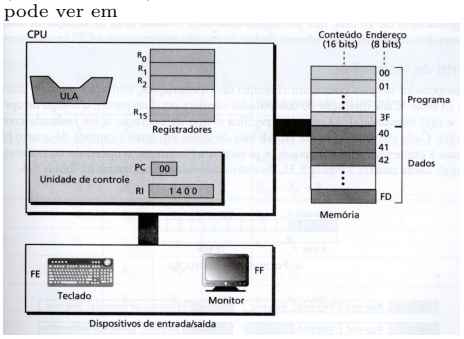
4

205-76551 - n lim

CEP - UP - UTFPR - PUCPr - UFPR –
10/06/2024 - 08:33:04.1
Prof Dr P Kantek (pkantek@gmail.com)
Um computador simples VIVXq06a, V: 1.31
76568 GUILHERME VITOR DA SILVA
24CC2205 - 8 27/jun limite 1. avaliacao _____ /
/ _____

Um computador simples

Vamos projetar um computador simples, para estudar de perto a sua arquitetura e seu processamento de instruções. O fato de ser um computador teórico (portanto irreal) não atrapalha. Projetado como se pode ver em



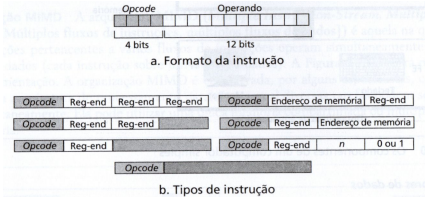
o computador terá 3 componentes: CPU, memória e subsistema de E/S. A CPU é dividida em 3 seções: registradores de dados, unidade aritmético-lógica e unidade de controle.

Existem 16 registradores de dados de 16 bits cada, e com endereços hexadecimais 0,1,...,F, conhecidos como $R_0, R_1, ..., R_{15}$. A unidade de controle controla as operações da ULA e os acessos à memória e ao subsistema de E/S. Tem 2 registradores dedicados: o PC (*Program Counter*) que aponta para o endereço de memória que contém a próxima instrução e o RI (*registrador de instrução*) mantém a instrução que está sendo executada no ciclo corrente. Após cada ciclo, o PC é incrementado (quase sempre) em 1 e o RI é atualizado com a próxima instrução.

A memória principal tem 256 localizações de 16 bits cada. Seus endereços binários variam entre 00000000_2 e 11111101_2 ou entre 00_{16} e FD_{16}. A memória contém dados e instruções. As primeiras 64 localizações (de 00 a $3F_{16}$ são dedicadas a instruções que são armazenadas em posições consecutivas. Já a área de 40 a FD_{16} são utilizadas para armazenar dados.

Já o subsistema de E/S é bastante primitivo, consistindo de um teclado e um vídeo. Assume-se que o teclado usa a posição de memória FE_{16} e o monitor de vídeo, a posição FF_{16} se comportando como registradores de 16 bits interagindo com a CPU como uma posição de memória qualquer.

Conjunto de instruções Existem 16 instruções, mas só vamos usar 14. Cada instrução tem 2 partes o *opcode* e o(s) operando(s). O opcode indica o que fazer com o operando. Assim, a instrução (de 16 bits) está dividida em 4 campos de 4 bits cada. O primeiro sub-campo é o opcode e os outros 12 são interpretados de acordo com o opcode. Veja em



Veja que nem todas as operações exigem três operandos. O que não é usado não precisa estar preenchido com zeros. Um endereço de registrador é indicado por um único dígito hexadecimal e usa um único sub-campo, enquanto um endereço de memória usa dois dígitos hexadecimais (portanto 2 sub-campos). Veja as instruções projetadas

Instrução	Código	Operandos			Ação
	d_1	d_2	d_3	d_4	
HALT	0				Para a execução do programa
LOAD	1	R_D	M_F		$R_D \leftarrow M_F$
STORE	2	M_D		R_F	$M_D \leftarrow R_F$
ADDI	3	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
ADDF	4	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
MOVE	5	R_D	R_F		$R_D \leftarrow R_F$
NOT	6	R_D	R_F		$R_D \leftarrow \overline{R}_F$
AND	7	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ AND R_{F_2}
OR	8	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ OR R_{F_2}
XOR	9	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ XOR R_{F_2}
INC	A	R			$R \leftarrow R + 1$
DEC	B	R			$R \leftarrow R - 1$
ROTATE	C	R	n	0 ou 1	Rot_n R
JUMP	D	R	n		Se $R_0 \neq$ R então PC = *n*, do co

Legenda: R_F, R_{F_1}, R_{F_2}: endereço hexadecimal dos registradores fonte
R_D: endereço hexadecimal do registrador destino
M_F: endereço hexadecimal da localização de memória fonte
M_D: endereço hexadecimal da localização da memória destino
n: número hexadecimal
d_1, d_2, d_3, d_4: primeiro, segundo, terceiro e quarto dígitos hexadecimais

Exemplo Acompanhe a execução da adição de dois inteiros A e B colocando o resultado em C. Em alto nível a instrução é $C = A + B$. Para isso, o computador precisa colocar os 2 operandos em registradores distintos (por exemplo R_0 e R_1) e determinar que o resultado vá para outro registrador (por exemplo R_2). A ULA só pode operar se os dados estiverem em registradores. Mas, como há um número limitado de registradores, há um constante fluxo de vai-e-vem entre eles e a memória. Por simplicidade assume-se que os dois valores estão em na posição 40_{16} e 41_{16} enquanto o resultado será colocado na posição 42. Desta maneira esta soma precisa 5 instruções:

1. Carrega o conteúdo de M_{40} em R_0
2. carrega o conteúdo de M_{41} em R_1
3. Adiciona R_0 com R_1 e coloca o resultado em R_2
4. Armazena R_2 em M_{42}
5. termina o programa

Veja o programa em linguagem de máquina

Código	Interpretação			
$(1040)_{16}$	1: LOAD	0: R_0	40: M_{40}	
$(1141)_{16}$	1: LOAD	1: R_1	41: M_{41}	
$(3201)_{16}$	3: ADDI	2: R_2	0: R_0	1: R_1
$(2422)_{16}$	2: STORE	42: M_{42}		2: R_2
$(0000)_{16}$	0: HALT			

Perceba que lá na máquina o programa é `1040114132012422 0000` (é mole ?).

Note que isto é só uma pequena parte da coisa. Antes, precisaríamos ter digitado um valor, que obviamente entraria na posição do teclado e transferido este valor para o endereço 40. Depois a mesma coisa indo para o endereço 41, depois a execução deste programa e finalmente a transferência do valor do endereço 42 para o monitor de vídeo. Ufa! Apenas para sua referência o programa completo é

$$\underbrace{1FFE}_{1}\underbrace{240F}_{2}\underbrace{1FFE}_{3}\underbrace{241F}_{4}\underbrace{1040}_{5}\underbrace{1141}_{6}\underbrace{3201}_{7}\underbrace{2422}_{8}$$

$$\underbrace{1F42}_{9}\underbrace{2FFF}_{10}\underbrace{0000}_{11}$$

As instruções 1 a 4 são para entrada e as 9 a 10 para saída.

Exemplo Nos exercicios a seguir, considere que os dados ja foram lidos nas posições de memória citadas e que serão exibidos a partir das posições também citadas. O que você deve resolver é a tradução das expressões aritméticas para linguagem de máquina e vice versa. Para facilitar a correção, use sempre o menor registrador disponível no momento.

Pede-se que você traduza para a linguagem de máquina a expressão $C \leftarrow A + (2 * B)$, sendo que &A=61, &B=65 e &D=72. Eis como ficaria

```
1065 - LOAD,0,65
3100 - ADDI,1,0,0  // 1=0+0
1261 - LOAD,2,61
3312 - ADDI,3,1,2  // 3=1+2
2723 - STORE,72,3
em linguagem de máquina 1065.3100.1261.3312.2723
```

Pede-se também o que faz o programa

```
1060.3100.3210.B200.B211.2732
```

A instrução `1060`, carrega no R_0 o conteúdo do endereço 60.
A instrução `3100`, soma os conteudos de R_0 e R_0 colocando o resultado em R_1, ou $R_1 = R_0 + R_0$
A instrução `3210`, soma os conteudos de R_1 e R_0 colocando o resultado em R_2, ou $R_2 = R_1 + R_0$ que devidamente interpretado com o anterior faz $R_2 = 3 \times R_0$
Depois, a instrução `B200`, decrementa o R_2. Note que o `00` que termina a instrução é puro lixo e não interessa. Outra maneira de representar esta instrução é `R2??` Mais um `B211`, que novamente decrementa o R_2. Agora o lixo é `11`
Finalmente, `2732` armazena o R_2 no endereço 73. Em resumo, aquele programa calcula $(X)_{73} \leftarrow ((Y)_{60} \times 3) - 2$

☞ Para você fazer

1. Traduza para linguagem de máquina a expressão $C = B - 1$,com &C=45,&B=60
2. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1040.1150.1260.3301.3332.2703
3. Traduza para linguagem de máquina a expressão $B = A + 3$,com &A=45,&B=60
4. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.3100.3101.3101.3101.2601

1

2

3

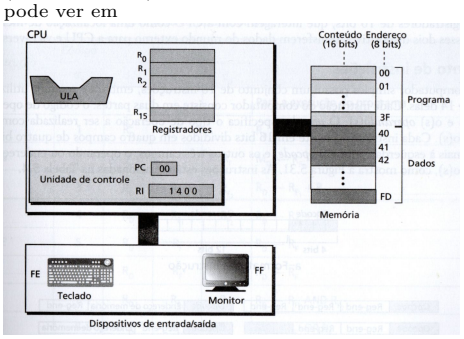
4

205-76568 - n lim

CEP - UP - UTFPR - PUCPr - UFPR – 10/06/2024 - 08:33:04.1
Prof Dr P Kantek (pkantek@gmail.com)
Um computador simples VIVXq06a, V: 1.31
76575 LAURA PACHECO DELATTRE
24CC2205 - 9 27/jun limite 1. avaliacao _____ / / _____

Um computador simples

Vamos projetar um computador simples, para estudar de perto a sua arquitetura e seu processamento de instruções. O fato de ser um computador teórico (portanto irreal) não atrapalha. Projetado como se pode ver em



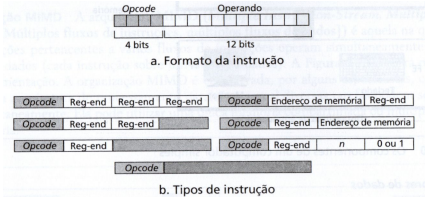
o computador terá 3 componentes: CPU, memória e subsistema de E/S. A CPU é dividida em 3 seções: registradores de dados, unidade aritmético-lógica e unidade de controle.

Existem 16 registradores de dados de 16 bits cada, e com endereços hexadecimais 0,1,...,F, conhecidos como $R_0, R_1, ..., R_{15}$. A unidade de controle controla as operações da ULA e os acessos à memória e ao subsistema de E/S. Tem 2 registradores dedicados: o PC (*Program Counter*) que aponta para o endereço de memória que contém a próxima instrução e o RI (*registrador de instrução*) mantém a instrução que está sendo executada no ciclo corrente. Após cada ciclo, o PC é incrementado (quase sempre) em 1 e o RI é atualizado com a próxima instrução.

A memória principal tem 256 localizações de 16 bits cada. Seus endereços binários variam entre 00000000_2 e 11111101_2 ou entre 00_{16} e FD_{16}. A memória contém dados e instruções. As primeiras 64 localizações (de 00 a $3F_{16}$ são dedicadas a instruções que são armazenadas em posições consecutivas. Já a área de 40 a FD_{16} são utilizadas para armazenar dados.

Já o subsistema de E/S é bastante primitivo, consistindo de um teclado e um vídeo. Assume-se que o teclado usa a posição de memória FE_{16} e o monitor de vídeo, a posição FF_{16} se comportando como registradores de 16 bits interagindo com a CPU como uma posição de memória qualquer.

Conjunto de instruções Existem 16 instruções, mas só vamos usar 14. Cada instrução tem 2 partes o *opcode* e o(s) operando(s). O opcode indica o que fazer com o operando. Assim, a instrução (de 16 bits) está dividida em 4 campos de 4 bits cada. O primeiro sub-campo é o opcode e os outros 12 são interpretados de acordo com o opcode. Veja em



Veja que nem todas as operações exigem três operandos. O que não é usado não precisa estar preenchido com zeros. Um endereço de registrador é indicado por um único dígito hexadecimal e usa um único sub-campo, enquanto um endereço de memória usa dois dígitos hexadecimais (portanto 2 sub-campos). Veja as instruções projetadas

Instrução	Código	Operandos			Ação
	d_1	d_2	d_3	d_4	
HALT	0				Para a execução do programa
LOAD	1	R_D	M_F		$R_D \leftarrow M_F$
STORE	2	M_D		R_F	$M_D \leftarrow R_F$
ADDI	3	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
ADDF	4	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
MOVE	5	R_D	R_F		$R_D \leftarrow R_F$
NOT	6	R_D	R_F		$R_D \leftarrow \overline{R}_F$
AND	7	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ AND R_{F_2}
OR	8	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ OR R_{F_2}
XOR	9	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ XOR R_{F_2}
INC	A	R			$R \leftarrow R + 1$
DEC	B	R			$R \leftarrow R - 1$
ROTATE	C	R	n	0 ou 1	Rot_n R
JUMP	D	R	n		Se $R_0 \neq$ R então PC = *n*, do co

Legenda: R_F, R_{F_1}, R_{F_2}: endereço hexadecimal dos registradores fonte
R_D: endereço hexadecimal do registrador destino
M_F: endereço hexadecimal da localização de memória fonte
M_D: endereço hexadecimal da localização da memória destino
n: número hexadecimal
d_1, d_2, d_3, d_4: primeiro, segundo, terceiro e quarto dígitos hexadecimais

Exemplo Acompanhe a execução da adição de dois inteiros A e B colocando o resultado em C. Em alto nível a instrução é $C = A + B$. Para isso, o computador precisa colocar os 2 operandos em registradores distintos (por exemplo R_0 e R_1) e determinar que o resultado vá para outro registrador (por exemplo R_2). A ULA só pode operar se os dados estiverem em registradores. Mas, como há um número limitado de registradores, há um constante fluxo de vai-e-vem entre eles e a memória. Por simplicidade assume-se que os dois valores estão em na posição 40_{16} e 41_{16} enquanto o resultado será colocado na posição 42. Desta maneira esta soma precisa 5 instruções:

1. Carrega o conteúdo de M_{40} em R_0
2. carrega o conteúdo de M_{41} em R_1
3. Adiciona R_0 com R_1 e coloca o resultado em R_2
4. Armazena R_2 em M_{42}
5. termina o programa

Veja o programa em linguagem de máquina

Código	Interpretação			
$(1040)_{16}$	1: LOAD	0: R_0	40: M_{40}	
$(1141)_{16}$	1: LOAD	1: R_1	41: M_{41}	
$(3201)_{16}$	3: ADDI	2: R_2	0: R_0	1: R_1
$(2422)_{16}$	2: STORE	42: M_{42}		2: R_2
$(0000)_{16}$	0: HALT			

Perceba que lá na máquina o programa é `1040114132012422 0000` (é mole ?).

Note que isto é só uma pequena parte da coisa. Antes, precisaríamos ter digitado um valor, que obviamente entraria na posição do teclado e transferido este valor para o endereço 40. Depois a mesma coisa indo para o endereço 41, depois a execução deste programa e finalmente a transferência do valor do endereço 42 para o monitor de vídeo. Ufa! Apenas para sua referência o programa completo é

$$\underbrace{1FFE}_{1}\underbrace{240F}_{2}\underbrace{1FFE}_{3}\underbrace{241F}_{4}\underbrace{1040}_{5}\underbrace{1141}_{6}\underbrace{3201}_{7}\underbrace{2422}_{8}$$

$$\underbrace{1F42}_{9}\underbrace{2FFF}_{10}\underbrace{0000}_{11}$$

As instruções 1 a 4 são para entrada e as 9 a 10 para saída.

Exemplo Nos exercicios a seguir, considere que os dados ja foram lidos nas posições de memória citadas e que serão exibidos a partir das posições também citadas. O que você deve resolver é a tradução das expressões aritméticas para linguagem de máquina e vice versa. Para facilitar a correção, use sempre o menor registrador disponível no momento.

Pede-se que você traduza para a linguagem de máquina a expressão $C \leftarrow A + (2 * B)$, sendo que &A=61, &B=65 e &D=72. Eis como ficaria

```
1065 - LOAD,0,65
3100 - ADDI,1,0,0  // 1=0+0
1261 - LOAD,2,61
3312 - ADDI,3,1,2  // 3=1+2
2723 - STORE,72,3
em linguagem de máquina 1065.3100.1261.3312.2723
```

Pede-se também o que faz o programa

```
1060.3100.3210.B200.B211.2732
```

A instrução `1060`, carrega no R_0 o conteúdo do endereço 60.
A instrução `3100`, soma os conteudos de R_0 e R_0 colocando o resultado em R_1, ou $R_1 = R_0 + R_0$
A instrução `3210`, soma os conteudos de R_1 e R_0 colocando o resultado em R_2, ou $R_2 = R_1 + R_0$ que devidamente interpretado com o anterior faz $R_2 = 3 \times R_0$
Depois, a instrução `B200`, decrementa o R_2. Note que o `00` que termina a instrução é puro lixo e não interessa. Outra maneira de representar esta instrução é `R2??` Mais um `B211`, que novamente decrementa o R_2. Agora o lixo é `11`
Finalmente, `2732` armazena o R_2 no endereço 73. Em resumo, aquele programa calcula $(X)_{73} \leftarrow ((Y)_{60} \times 3) - 2$

☞ Para você fazer

1. Traduza para linguagem de máquina a expressão $C = B + 2$,com &C=45,&B=60
2. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1060.B0??.2450
3. Traduza para linguagem de máquina a expressão $D = B + 1$,com &B=45,&D=70
4. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.B0??.B0??.2600

1

2

3

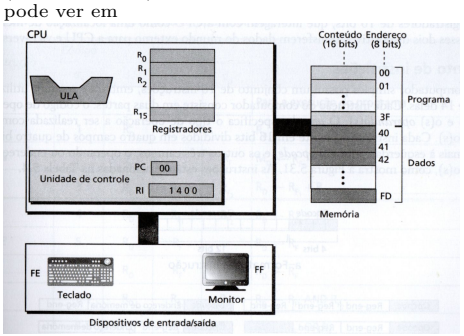
4

205-76575 - n lim

CEP - UP - UTFPR - PUCPr - UFPR –
10/06/2024 - 08:33:04.1
Prof Dr P Kantek (pkantek@gmail.com)
Um computador simples VIVXq06a, V: 1.31 76582
LUIZ MIGUEL OLINEK 24CC2205 - 10 27/jun
limite 1. avaliacao _____ / _____ / _____

Um computador simples

Vamos projetar um computador simples, para estudar de perto a sua arquitetura e seu processamento de instruções. O fato de ser um computador teórico (portanto irreal) não atrapalha. Projetado como se pode ver em



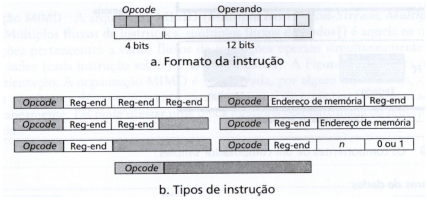
o computador terá 3 componentes: CPU, memória e subsistema de E/S. A CPU é dividida em 3 seções: registradores de dados, unidade aritmético-lógica e unidade de controle.

Existem 16 registradores de dados de 16 bits cada, e com endereços hexadecimais 0,1,...,F, conhecidos como $R_0, R_1, ..., R_{15}$. A unidade de controle controla as operações da ULA e os acessos à memória e ao subsistema de E/S. Tem 2 registradores dedicados: o PC (*Program Counter*) que aponta para o endereço de memória que contém a próxima instrução e o RI (*registrador de instrução*) mantém a instrução que está sendo executada no ciclo corrente. Após cada ciclo, o PC é incrementado (quase sempre) em 1 e o RI é atualizado com a próxima instrução.

A memória principal tem 256 localizações de 16 bits cada. Seus endereços binários variam entre 00000000_2 e 11111101_2 ou entre 00_{16} e FD_{16}. A memória contém dados e instruções. As primeiras 64 localizações (de 00 a $3F_{16}$ são dedicadas a instruções que são armazenadas em posições consecutivas. Já a área de 40 a FD_{16} são utilizadas para armazenar dados.

Já o subsistema de E/S é bastante primitivo, consistindo de um teclado e um vídeo. Assume-se que o teclado usa a posição de memória FE_{16} e o monitor de vídeo, a posição FF_{16} se comportando como registradores de 16 bits interagindo com a CPU como uma posição de memória qualquer.

Conjunto de instruções Existem 16 instruções, mas só vamos usar 14. Cada instrução tem 2 partes o *opcode* e o(s) operando(s). O opcode indica o que fazer com o operando. Assim, a instrução (de 16 bits) está dividida em 4 campos de 4 bits cada. O primeiro sub-campo é o opcode e os outros 12 são interpretados de acordo com o opcode. Veja em



Veja que nem todas as operações exigem três operandos. O que não é usado não precisa estar preenchido com zeros. Um endereço de registrador é indicado por um único dígito hexadecimal e usa um único sub-campo, enquanto um endereço de memória usa dois dígitos hexadecimais (portanto 2 sub-campos). Veja as instruções projetadas

Instrução	Código	Operandos			Ação
	d_1	d_2	d_3	d_4	
HALT	0				Para a execução do programa
LOAD	1	R_D	M_F		$R_D \leftarrow M_F$
STORE	2	M_D		R_F	$M_D \leftarrow R_F$
ADDI	3	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
ADDF	4	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
MOVE	5	R_D	R_F		$R_D \leftarrow R_F$
NOT	6	R_D	R_F		$R_D \leftarrow \overline{R}_F$
AND	7	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ AND R_{F_2}
OR	8	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ OR R_{F_2}
XOR	9	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ XOR R_{F_2}
INC	A	R			$R \leftarrow R + 1$
DEC	B	R			$R \leftarrow R - 1$
ROTATE	C	R	n	0 ou 1	Rot_n R
JUMP	D	R	n		Se $R_0 \neq$ R então PC = *n*, do co

Legenda: R_F, R_{F_1}, R_{F_2}: endereço hexadecimal dos registradores fonte
R_D: endereço hexadecimal do registrador destino
M_F: endereço hexadecimal da localização de memória fonte
M_D: endereço hexadecimal da localização da memória destino
n: número hexadecimal
d_1, d_2, d_3, d_4: primeiro, segundo, terceiro e quarto dígitos hexadecimais

Exemplo Acompanhe a execução da adição de dois inteiros A e B colocando o resultado em C. Em alto nível a instrução é $C = A + B$. Para isso, o computador precisa colocar os 2 operandos em registradores distintos (por exemplo R_0 e R_1) e determinar que o resultado vá para outro registrador (por exemplo R_2). A ULA só pode operar se os dados estiverem em registradores. Mas, como há um número limitado de registradores, há um constante fluxo de vai-e-vem entre eles e a memória. Por simplicidade assume-se que os dois valores estão em na posição 40_{16} e 41_{16} enquanto o resultado será colocado na posição 42. Desta maneira esta soma precisa 5 instruções:

1. Carrega o conteúdo de M_{40} em R_0
2. carrega o conteúdo de M_{41} em R_1
3. Adiciona R_0 com R_1 e coloca o resultado em R_2
4. Armazena R_2 em M_{42}
5. termina o programa

Veja o programa em linguagem de máquina

Código	Interpretação			
$(1040)_{16}$	1: LOAD	0: R_0	40: M_{40}	
$(1141)_{16}$	1: LOAD	1: R_1	41: M_{41}	
$(3201)_{16}$	3: ADDI	2: R_2	0: R_0	1: R_1
$(2422)_{16}$	2: STORE	42: M_{42}		2: R_2
$(0000)_{16}$	0: HALT			

Perceba que lá na máquina o programa é `1040114132012422 0000` (é mole ?).

Note que isto é só uma pequena parte da coisa. Antes, precisaríamos ter digitado um valor, que obviamente entraria na posição do teclado e transferido este valor para o endereço 40. Depois a mesma coisa indo para o endereço 41, depois a execução deste programa e finalmente a transferência do valor do endereço 42 para o monitor de vídeo. Ufa! Apenas para sua referência o programa completo é

$$\underbrace{1FFE}_{1}\ \underbrace{240F}_{2}\ \underbrace{1FFE}_{3}\ \underbrace{241F}_{4}\ \underbrace{1040}_{5}\ \underbrace{1141}_{6}\ \underbrace{3201}_{7}\ \underbrace{2422}_{8}$$

$$\underbrace{1F42}_{9}\ \underbrace{2FFF}_{10}\ \underbrace{0000}_{11}$$

As instruções 1 a 4 são para entrada e as 9 a 10 para saída.

Exemplo Nos exercicios a seguir, considere que os dados ja foram lidos nas posições de memória citadas e que serão exibidos a partir das posições também citadas. O que você deve resolver é a tradução das expressões aritméticas para linguagem de máquina e vice versa. Para facilitar a correção, use sempre o menor registrador disponível no momento.

Pede-se que você traduza para a linguagem de máquina a expressão $C \leftarrow A + (2 * B)$, sendo que &A=61, &B=65 e &D=72. Eis como ficaria

```
1065 - LOAD,0,65
3100 - ADDI,1,0,0  // 1=0+0
1261 - LOAD,2,61
3312 - ADDI,3,1,2  // 3=1+2
2723 - STORE,72,3
em linguagem de máquina 1065.3100.1261.3312.2723
```

Pede-se também o que faz o programa

```
1060.3100.3210.B200.B211.2732
```

A instrução `1060`, carrega no R_0 o conteúdo do endereço 60.
A instrução `3100`, soma os conteudos de R_0 e R_0 colocando o resultado em R_1, ou $R_1 = R_0 + R_0$
A instrução `3210`, soma os conteudos de R_1 e R_0 colocando o resultado em R_2, ou $R_2 = R_1 + R_0$ que devidamente interpretado com o anterior faz $R_2 = 3 \times R_0$
Depois, a instrução `B200`, decrementa o R_2. Note que o `00` que termina a instrução é puro lixo e não interessa. Outra maneira de representar esta instrução é `R2??` Mais um `B211`, que novamente decrementa o R_2. Agora o lixo é `11`
Finalmente, `2732` armazena o R_2 no endereço 73. Em resumo, aquele programa calcula $(X)_{73} \leftarrow ((Y)_{60} \times 3) - 2$

☞ Para você fazer

1. Traduza para linguagem de máquina a expressão $B = A + 3$,com &A=45,&B=60
2. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1040.1150.1260.3301.3332.2703
3. Traduza para linguagem de máquina a expressão $D = A+A+A+A+A$, &A=45,&D=60
4. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1060.A0??.A0??.2450

1

2

3

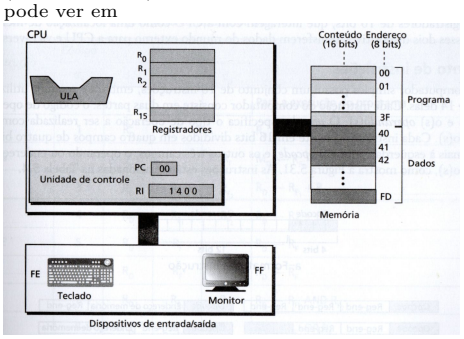
4

205-76582 - n lim

CEP - UP - UTFPR - PUCPr - UFPR –
10/06/2024 - 08:33:04.1
Prof Dr P Kantek (pkantek@gmail.com)
Um computador simples VIVXq06a, V: 1.31
76599 NICOLE DA SILVA DE AZEVEDO
24CC2205 - 11 27/jun limite 1. avaliacao _____ /
/ _____

Um computador simples

Vamos projetar um computador simples, para estudar de perto a sua arquitetura e seu processamento de instruções. O fato de ser um computador teórico (portanto irreal) não atrapalha. Projetado como se pode ver em



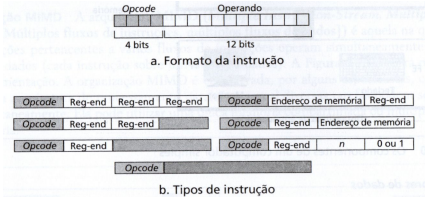
o computador terá 3 componentes: CPU, memória e subsistema de E/S. A CPU é dividida em 3 seções: registradores de dados, unidade aritmético-lógica e unidade de controle.

Existem 16 registradores de dados de 16 bits cada, e com endereços hexadecimais 0,1,...,F, conhecidos como $R_0, R_1, ..., R_{15}$. A unidade de controle controla as operações da ULA e os acessos à memória e ao subsistema de E/S. Tem 2 registradores dedicados: o PC (*Program Counter*) que aponta para o endereço de memória que contém a próxima instrução e o RI (*registrador de instrução*) mantém a instrução que está sendo executada no ciclo corrente. Após cada ciclo, o PC é incrementado (quase sempre) em 1 e o RI é atualizado com a próxima instrução.

A memória principal tem 256 localizações de 16 bits cada. Seus endereços binários variam entre 00000000_2 e 11111101_2 ou entre 00_{16} e FD_{16}. A memória contém dados e instruções. As primeiras 64 localizações (de 00 a $3F_{16}$ são dedicadas a instruções que são armazenadas em posições consecutivas. Já a área de 40 a FD_{16} são utilizadas para armazenar dados.

Já o subsistema de E/S é bastante primitivo, consistindo de um teclado e um vídeo. Assume-se que o teclado usa a posição de memória FE_{16} e o monitor de vídeo, a posição FF_{16} se comportando como registradores de 16 bits interagindo com a CPU como uma posição de memória qualquer.

Conjunto de instruções Existem 16 instruções, mas só vamos usar 14. Cada instrução tem 2 partes o *opcode* e o(s) operando(s). O opcode indica o que fazer com o operando. Assim, a instrução (de 16 bits) está dividida em 4 campos de 4 bits cada. O primeiro sub-campo é o opcode e os outros 12 são interpretados de acordo com o opcode. Veja em



Veja que nem todas as operações exigem três operandos. O que não é usado não precisa estar preenchido com zeros. Um endereço de registrador é indicado por um único dígito hexadecimal e usa um único sub-campo, enquanto um endereço de memória usa dois dígitos hexadecimais (portanto 2 sub-campos). Veja as instruções projetadas

Instrução	Código	Operandos			Ação
	d_1	d_2	d_3	d_4	
HALT	0				Para a execução do programa
LOAD	1	R_D	M_F		$R_D \leftarrow M_F$
STORE	2	M_D		R_F	$M_D \leftarrow R_F$
ADDI	3	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
ADDF	4	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
MOVE	5	R_D	R_F		$R_D \leftarrow R_F$
NOT	6	R_D	R_F		$R_D \leftarrow \overline{R}_F$
AND	7	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ AND R_{F_2}
OR	8	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ OR R_{F_2}
XOR	9	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ XOR R_{F_2}
INC	A	R			$R \leftarrow R + 1$
DEC	B	R			$R \leftarrow R - 1$
ROTATE	C	R	n	0 ou 1	Rot_n R
JUMP	D	R	n		Se $R_0 \neq$ R então PC = *n*, do co

Legenda: R_F, R_{F_1}, R_{F_2}: endereço hexadecimal dos registradores fonte
R_D: endereço hexadecimal do registrador destino
M_F: endereço hexadecimal da localização de memória fonte
M_D: endereço hexadecimal da localização da memória destino
n: número hexadecimal
d_1, d_2, d_3, d_4: primeiro, segundo, terceiro e quarto dígitos hexadecimais

Exemplo Acompanhe a execução da adição de dois inteiros A e B colocando o resultado em C. Em alto nível a instrução é $C = A + B$. Para isso, o computador precisa colocar os 2 operandos em registradores distintos (por exemplo R_0 e R_1) e determinar que o resultado vá para outro registrador (por exemplo R_2). A ULA só pode operar se os dados estiverem em registradores. Mas, como há um número limitado de registradores, há um constante fluxo de vai-e-vem entre eles e a memória. Por simplicidade assume-se que os dois valores estão em na posição 40_{16} e 41_{16} enquanto o resultado será colocado na posição 42. Desta maneira esta soma precisa 5 instruções:

1. Carrega o conteúdo de M_{40} em R_0
2. carrega o conteúdo de M_{41} em R_1
3. Adiciona R_0 com R_1 e coloca o resultado em R_2
4. Armazena R_2 em M_{42}
5. termina o programa

Veja o programa em linguagem de máquina

Código	Interpretação			
$(1040)_{16}$	1: LOAD	0: R_0	40: M_{40}	
$(1141)_{16}$	1: LOAD	1: R_1	41: M_{41}	
$(3201)_{16}$	3: ADDI	2: R_2	0: R_0	1: R_1
$(2422)_{16}$	2: STORE	42: M_{42}		2: R_2
$(0000)_{16}$	0: HALT			

Perceba que lá na máquina o programa é `1040114132012422000`0 (é mole ?).

Note que isto é só uma pequena parte da coisa. Antes, precisaríamos ter digitado um valor, que obviamente entraria na posição do teclado e transferido este valor para o endereço 40. Depois a mesma coisa indo para o endereço 41, depois a execução deste programa e finalmente a transferência do valor do endereço 42 para o monitor de vídeo. Ufa! Apenas para sua referência o programa completo é

$$\underbrace{1FFE}_{1}\ \underbrace{240F}_{2}\ \underbrace{1FFE}_{3}\ \underbrace{241F}_{4}\ \underbrace{1040}_{5}\ \underbrace{1141}_{6}\ \underbrace{3201}_{7}\ \underbrace{2422}_{8}$$

$$\underbrace{1F42}_{9}\ \underbrace{2FFF}_{10}\ \underbrace{0000}_{11}$$

As instruções 1 a 4 são para entrada e as 9 a 10 para saída.

Exemplo Nos exercicios a seguir, considere que os dados ja foram lidos nas posições de memória citadas e que serão exibidos a partir das posições também citadas. O que você deve resolver é a tradução das expressões aritméticas para linguagem de máquina e vice versa. Para facilitar a correção, use sempre o menor registrador disponível no momento.

Pede-se que você traduza para a linguagem de máquina a expressão $C \leftarrow A + (2 * B)$, sendo que &A=61, &B=65 e &D=72. Eis como ficaria

```
1065 - LOAD,0,65
3100 - ADDI,1,0,0  // 1=0+0
1261 - LOAD,2,61
3312 - ADDI,3,1,2  // 3=1+2
2723 - STORE,72,3
em linguagem de máquina 1065.3100.1261.3312.2723
```

Pede-se também o que faz o programa

```
1060.3100.3210.B200.B211.2732
```

A instrução `1060`, carrega no R_0 o conteúdo do endereço 60.
A instrução `3100`, soma os conteudos de R_0 e R_0 colocando o resultado em R_1, ou $R_1 = R_0 + R_0$
A instrução `3210`, soma os conteudos de R_1 e R_0 colocando o resultado em R_2, ou $R_2 = R_1 + R_0$ que devidamente interpretado com o anterior faz $R_2 = 3 \times R_0$
Depois, a instrução `B200`, decrementa o R_2. Note que o `00` que termina a instrução é puro lixo e não interessa. Outra maneira de representar esta instrução é `R2??` Mais um `B211`, que novamente decrementa o R_2. Agora o lixo é `11`
Finalmente, `2732` armazena o R_2 no endereço 73. Em resumo, aquele programa calcula $(X)_{73} \leftarrow ((Y)_{60} \times 3) - 2$

☞ Para você fazer

1. Traduza para linguagem de máquina a expressão $C = B - 1$,com &C=45,&B=60
2. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.3100.3101.3101.3101.2601
3. Traduza para linguagem de máquina a expressão $B = A - 2$,com &A=45,&B=60
4. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.A0??.A0??.A0??.2600

1

2

3

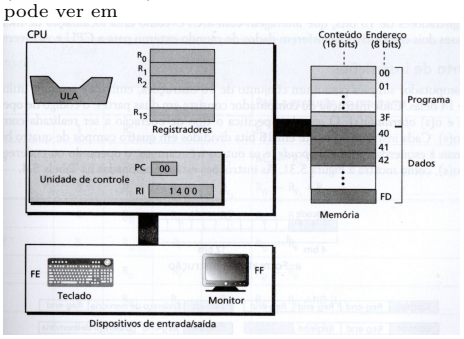
4

205-76599 - n lim

CEP - UP - UTFPR - PUCPr - UFPR –
10/06/2024 - 08:33:04.1
Prof Dr P Kantek (pkantek@gmail.com)
Um computador simples VIVXq06a, V: 1.31
76601 PAULO JOSE FREIRE CORREA
24CC2205 - 12 27/jun limite 1. avaliacao _____ /
/ _____

Um computador simples

Vamos projetar um computador simples, para estudar de perto a sua arquitetura e seu processamento de instruções. O fato de ser um computador teórico (portanto irreal) não atrapalha. Projetado como se pode ver em



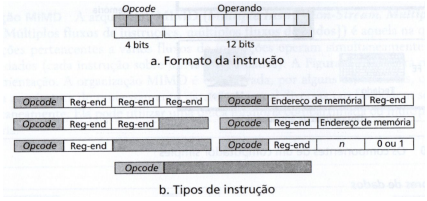
o computador terá 3 componentes: CPU, memória e subsistema de E/S. A CPU é dividida em 3 seções: registradores de dados, unidade aritmético-lógica e unidade de controle.

Existem 16 registradores de dados de 16 bits cada, e com endereços hexadecimais 0,1,...,F, conhecidos como $R_0, R_1, ..., R_{15}$. A unidade de controle controla as operações da ULA e os acessos à memória e ao subsistema de E/S. Tem 2 registradores dedicados: o PC (*Program Counter*) que aponta para o endereço de memória que contém a próxima instrução e o RI (*registrador de instrução*) mantém a instrução que está sendo executada no ciclo corrente. Após cada ciclo, o PC é incrementado (quase sempre) em 1 e o RI é atualizado com a próxima instrução.

A memória principal tem 256 localizações de 16 bits cada. Seus endereços binários variam entre 00000000_2 e 11111101_2 ou entre 00_{16} e FD_{16}. A memória contém dados e instruções. As primeiras 64 localizações (de 00 a $3F_{16}$ são dedicadas a instruções que são armazenadas em posições consecutivas. Já a área de 40 a FD_{16} são utilizadas para armazenar dados.

Já o subsistema de E/S é bastante primitivo, consistindo de um teclado e um vídeo. Assume-se que o teclado usa a posição de memória FE_{16} e o monitor de vídeo, a posição FF_{16} se comportando como registradores de 16 bits interagindo com a CPU como uma posição de memória qualquer.

Conjunto de instruções Existem 16 instruções, mas só vamos usar 14. Cada instrução tem 2 partes o *opcode* e o(s) operando(s). O opcode indica o que fazer com o operando. Assim, a instrução (de 16 bits) está dividida em 4 campos de 4 bits cada. O primeiro sub-campo é o opcode e os outros 12 são interpretados de acordo com o opcode. Veja em



Veja que nem todas as operações exigem três operandos. O que não é usado não precisa estar preenchido com zeros. Um endereço de registrador é indicado por um único dígito hexadecimal e usa um único sub-campo, enquanto um endereço de memória usa dois dígitos hexadecimais (portanto 2 sub-campos). Veja as instruções projetadas

Instrução	Código	Operandos			Ação
	d_1	d_2	d_3	d_4	
HALT	0				Para a execução do programa
LOAD	1	R_D	M_F		$R_D \leftarrow M_F$
STORE	2	M_D		R_F	$M_D \leftarrow R_F$
ADDI	3	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
ADDF	4	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
MOVE	5	R_D	R_F		$R_D \leftarrow R_F$
NOT	6	R_D	R_F		$R_D \leftarrow \overline{R}_F$
AND	7	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ AND R_{F_2}
OR	8	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ OR R_{F_2}
XOR	9	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ XOR R_{F_2}
INC	A	R			$R \leftarrow R + 1$
DEC	B	R			$R \leftarrow R - 1$
ROTATE	C	R	n	0 ou 1	Rot_n R
JUMP	D	R	n		Se $R_0 \neq$ R então PC = *n*, do co

Legenda: R_F, R_{F_1}, R_{F_2}: endereço hexadecimal dos registradores fonte
R_D: endereço hexadecimal do registrador destino
M_F: endereço hexadecimal da localização de memória fonte
M_D: endereço hexadecimal da localização da memória destino
n: número hexadecimal
d_1, d_2, d_3, d_4: primeiro, segundo, terceiro e quarto dígitos hexadecimais

Exemplo Acompanhe a execução da adição de dois inteiros A e B colocando o resultado em C. Em alto nível a instrução é $C = A + B$. Para isso, o computador precisa colocar os 2 operandos em registradores distintos (por exemplo R_0 e R_1) e determinar que o resultado vá para outro registrador (por exemplo R_2). A ULA só pode operar se os dados estiverem em registradores. Mas, como há um número limitado de registradores, há um constante fluxo de vai-e-vem entre eles e a memória. Por simplicidade assume-se que os dois valores estão em na posição 40_{16} e 41_{16} enquanto o resultado será colocado na posição 42. Desta maneira esta soma precisa 5 instruções:

1. Carrega o conteúdo de M_{40} em R_0
2. carrega o conteúdo de M_{41} em R_1
3. Adiciona R_0 com R_1 e coloca o resultado em R_2
4. Armazena R_2 em M_{42}
5. termina o programa

Veja o programa em linguagem de máquina

Código	Interpretação			
$(1040)_{16}$	1: LOAD	0: R_0	40: M_{40}	
$(1141)_{16}$	1: LOAD	1: R_1	41: M_{41}	
$(3201)_{16}$	3: ADDI	2: R_2	0: R_0	1: R_1
$(2422)_{16}$	2: STORE	42: M_{42}		2: R_2
$(0000)_{16}$	0: HALT			

Perceba que lá na máquina o programa é `1040114132012422 0000` (é mole ?).

Note que isto é só uma pequena parte da coisa. Antes, precisaríamos ter digitado um valor, que obviamente entraria na posição do teclado e transferido este valor para o endereço 40. Depois a mesma coisa indo para o endereço 41, depois a execução deste programa e finalmente a transferência do valor do endereço 42 para o monitor de vídeo. Ufa! Apenas para sua referência o programa completo é

$$\underbrace{1FFE}_{1}\ \underbrace{240F}_{2}\ \underbrace{1FFE}_{3}\ \underbrace{241F}_{4}\ \underbrace{1040}_{5}\ \underbrace{1141}_{6}\ \underbrace{3201}_{7}\ \underbrace{2422}_{8}$$

$$\underbrace{1F42}_{9}\ \underbrace{2FFF}_{10}\ \underbrace{0000}_{11}$$

As instruções 1 a 4 são para entrada e as 9 a 10 para saída.

Exemplo Nos exercicios a seguir, considere que os dados ja foram lidos nas posições de memória citadas e que serão exibidos a partir das posições também citadas. O que você deve resolver é a tradução das expressões aritméticas para linguagem de máquina e vice versa. Para facilitar a correção, use sempre o menor registrador disponível no momento.

Pede-se que você traduza para a linguagem de máquina a expressão $C \leftarrow A + (2 * B)$, sendo que &A=61, &B=65 e &D=72. Eis como ficaria

```
1065 - LOAD,0,65
3100 - ADDI,1,0,0  // 1=0+0
1261 - LOAD,2,61
3312 - ADDI,3,1,2  // 3=1+2
2723 - STORE,72,3
em linguagem de máquina 1065.3100.1261.3312.2723
```

Pede-se também o que faz o programa

```
1060.3100.3210.B200.B211.2732
```

A instrução `1060`, carrega no R_0 o conteúdo do endereço 60.
A instrução `3100`, soma os conteudos de R_0 e R_0 colocando o resultado em R_1, ou $R_1 = R_0 + R_0$
A instrução `3210`, soma os conteudos de R_1 e R_0 colocando o resultado em R_2, ou $R_2 = R_1 + R_0$ que devidamente interpretado com o anterior faz $R_2 = 3 \times R_0$
Depois, a instrução `B200`, decrementa o R_2. Note que o `00` que termina a instrução é puro lixo e não interessa. Outra maneira de representar esta instrução é `R2??` Mais um `B211`, que novamente decrementa o R_2. Agora o lixo é `11`
Finalmente, `2732` armazena o R_2 no endereço 73. Em resumo, aquele programa calcula $(X)_{73} \leftarrow ((Y)_{60} \times 3) - 2$

☞ Para você fazer

1. Traduza para linguagem de máquina a expressão $D = A + A + A + A + A$, &A=45,&D=60
2. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.3100.2701
3. Traduza para linguagem de máquina a expressão $C = A * 3$,com &A=45,&C=70
4. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.A0??.2700

1

2

3

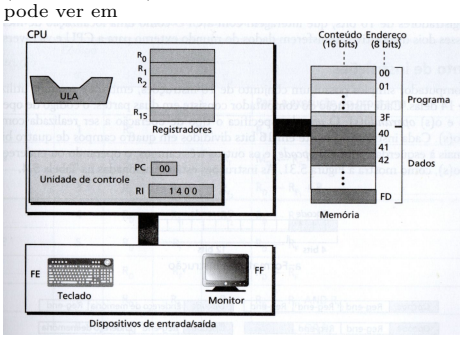
4

205-76601 - n lim

CEP - UP - UTFPR - PUCPr - UFPR – 10/06/2024 - 08:33:04.1
Prof Dr P Kantek (pkantek@gmail.com)
Um computador simples VIVXq06a, V: 1.31
76618 PEDRO HENRIQUE PIEKARSKI DE OL 24CC2205 - 13 27/jun limite 1. avaliacao _____ / / _____

Um computador simples

Vamos projetar um computador simples, para estudar de perto a sua arquitetura e seu processamento de instruções. O fato de ser um computador teórico (portanto irreal) não atrapalha. Projetado como se pode ver em



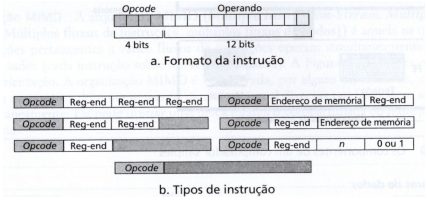
o computador terá 3 componentes: CPU, memória e subsistema de E/S. A CPU é dividida em 3 seções: registradores de dados, unidade aritmético-lógica e unidade de controle.

Existem 16 registradores de dados de 16 bits cada, e com endereços hexadecimais 0,1,...,F, conhecidos como $R_0, R_1, ..., R_{15}$. A unidade de controle controla as operações da ULA e os acessos à memória e ao subsistema de E/S. Tem 2 registradores dedicados: o PC (*Program Counter*) que aponta para o endereço de memória que contém a próxima instrução e o RI (*registrador de instrução*) mantém a instrução que está sendo executada no ciclo corrente. Após cada ciclo, o PC é incrementado (quase sempre) em 1 e o RI é atualizado com a próxima instrução.

A memória principal tem 256 localizações de 16 bits cada. Seus endereços binários variam entre 00000000_2 e 11111101_2 ou entre 00_{16} e FD_{16}. A memória contém dados e instruções. As primeiras 64 localizações (de 00 a $3F_{16}$ são dedicadas a instruções que são armazenadas em posições consecutivas. Já a área de 40 a FD_{16} são utilizadas para armazenar dados.

Já o subsistema de E/S é bastante primitivo, consistindo de um teclado e um vídeo. Assume-se que o teclado usa a posição de memória FE_{16} e o monitor de vídeo, a posição FF_{16} se comportando como registradores de 16 bits interagindo com a CPU como uma posição de memória qualquer.

Conjunto de instruções Existem 16 instruções, mas só vamos usar 14. Cada instrução tem 2 partes o *opcode* e o(s) operando(s). O opcode indica o que fazer com o operando. Assim, a instrução (de 16 bits) está dividida em 4 campos de 4 bits cada. O primeiro sub-campo é o opcode e os outros 12 são interpretados de acordo com o opcode. Veja em



Veja que nem todas as operações exigem três operandos. O que não é usado não precisa estar preenchido com zeros. Um endereço de registrador é indicado por um único dígito hexadecimal e usa um único sub-campo, enquanto um endereço de memória usa dois dígitos hexadecimais (portanto 2 sub-campos). Veja as instruções projetadas

Instrução	Código	Operandos			Ação
	d_1	d_2	d_3	d_4	
HALT	0				Para a execução do programa
LOAD	1	R_D	M_F		$R_D \leftarrow M_F$
STORE	2	M_D		R_F	$M_D \leftarrow R_F$
ADDI	3	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
ADDF	4	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
MOVE	5	R_D	R_F		$R_D \leftarrow R_F$
NOT	6	R_D	R_F		$R_D \leftarrow \overline{R}_F$
AND	7	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ AND R_{F_2}
OR	8	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ OR R_{F_2}
XOR	9	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ XOR R_{F_2}
INC	A	R			$R \leftarrow R + 1$
DEC	B	R			$R \leftarrow R - 1$
ROTATE	C	R	n	0 ou 1	Rot_n R
JUMP	D	R	n		Se $R_0 \neq$ R então PC = *n*, do co

Legenda: R_F, R_{F_1}, R_{F_2}: endereço hexadecimal dos registradores fonte
R_D: endereço hexadecimal do registrador destino
M_F: endereço hexadecimal da localização de memória fonte
M_D: endereço hexadecimal da localização da memória destino
n: número hexadecimal
d_1, d_2, d_3, d_4: primeiro, segundo, terceiro e quarto dígitos hexadecimais

Exemplo Acompanhe a execução da adição de dois inteiros A e B colocando o resultado em C. Em alto nível a instrução é $C = A + B$. Para isso, o computador precisa colocar os 2 operandos em registradores distintos (por exemplo R_0 e R_1) e determinar que o resultado vá para outro registrador (por exemplo R_2). A ULA só pode operar se os dados estiverem em registradores. Mas, como há um número limitado de registradores, há um constante fluxo de vai-e-vem entre eles e a memória. Por simplicidade assume-se que os dois valores estão em na posição 40_{16} e 41_{16} enquanto o resultado será colocado na posição 42. Desta maneira esta soma precisa 5 instruções:

1. Carrega o conteúdo de M_{40} em R_0
2. carrega o conteúdo de M_{41} em R_1
3. Adiciona R_0 com R_1 e coloca o resultado em R_2
4. Armazena R_2 em M_{42}
5. termina o programa

Veja o programa em linguagem de máquina

Código	Interpretação			
$(1040)_{16}$	1: LOAD	0: R_0	40: M_{40}	
$(1141)_{16}$	1: LOAD	1: R_1	41: M_{41}	
$(3201)_{16}$	3: ADDI	2: R_2	0: R_0	1: R_1
$(2422)_{16}$	2: STORE	42: M_{42}		2: R_2
$(0000)_{16}$	0: HALT			

Perceba que lá na máquina o programa é `1040114132012422000O` (é mole ?).

Note que isto é só uma pequena parte da coisa. Antes, precisaríamos ter digitado um valor, que obviamente entraria na posição do teclado e transferido este valor para o endereço 40. Depois a mesma coisa indo para o endereço 41, depois a execução deste programa e finalmente a transferência do valor do endereço 42 para o monitor de vídeo. Ufa! Apenas para sua referência o programa completo é

$$\underbrace{1FFE}_{1}\underbrace{240F}_{2}\underbrace{1FFE}_{3}\underbrace{241F}_{4}\underbrace{1040}_{5}\underbrace{1141}_{6}\underbrace{3201}_{7}\underbrace{2422}_{8}$$

$$\underbrace{1F42}_{9}\underbrace{2FFF}_{10}\underbrace{0000}_{11}$$

As instruções 1 a 4 são para entrada e as 9 a 10 para saída.

Exemplo Nos exercicios a seguir, considere que os dados ja foram lidos nas posições de memória citadas e que serão exibidos a partir das posições também citadas. O que você deve resolver é a tradução das expressões aritméticas para linguagem de máquina e vice versa. Para facilitar a correção, use sempre o menor registrador disponível no momento.

Pede-se que você traduza para a linguagem de máquina a expressão $C \leftarrow A + (2 * B)$, sendo que &A=61, &B=65 e &D=72. Eis como ficaria

```
1065 - LOAD,0,65
3100 - ADDI,1,0,0  // 1=0+0
1261 - LOAD,2,61
3312 - ADDI,3,1,2  // 3=1+2
2723 - STORE,72,3
em linguagem de máquina 1065.3100.1261.3312.2723
```

Pede-se também o que faz o programa

```
1060.3100.3210.B200.B211.2732
```

A instrução `1060`, carrega no R_0 o conteúdo do endereço 60.
A instrução `3100`, soma os conteudos de R_0 e R_0 colocando o resultado em R_1, ou $R_1 = R_0 + R_0$
A instrução `3210`, soma os conteudos de R_1 e R_0 colocando o resultado em R_2, ou $R_2 = R_1 + R_0$ que devidamente interpretado com o anterior faz $R_2 = 3 \times R_0$
Depois, a instrução `B200`, decrementa o R_2. Note que o `00` que termina a instrução é puro lixo e não interessa. Outra maneira de representar esta instrução é `R2??` Mais um `B211`, que novamente decrementa o R_2. Agora o lixo é `11`
Finalmente, `2732` armazena o R_2 no endereço 73. Em resumo, aquele programa calcula $(X)_{73} \leftarrow ((Y)_{60} \times 3) - 2$

☞ Para você fazer

1. Traduza para linguagem de máquina a expressão $C = B + 2$,com &C=45,&B=60
2. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.3100.3101.2701
3. Traduza para linguagem de máquina a expressão $D = A + B + C$,com &A=40,&B=50,&C=60,&D=70
4. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.A0??.2700

1

2

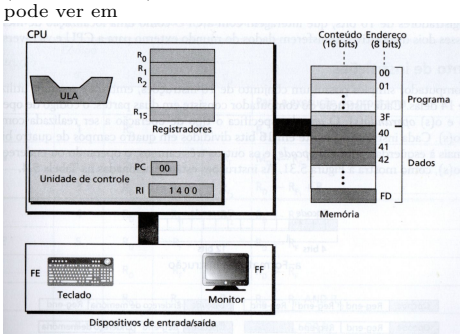
3

4

CEP - UP - UTFPR - PUCPr - UFPR –
10/06/2024 - 08:33:04.1
Prof Dr P Kantek (pkantek@gmail.com)
Um computador simples VIVXq06a, V: 1.31 76625
RAUL KNAPKI CUNHA 24CC2205 - 14 27/jun
limite 1. avaliacao _____ / _____ / _____

Um computador simples

Vamos projetar um computador simples, para estudar de perto a sua arquitetura e seu processamento de instruções. O fato de ser um computador teórico (portanto irreal) não atrapalha. Projetado como se pode ver em



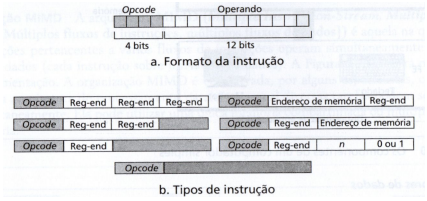
o computador terá 3 componentes: CPU, memória e subsistema de E/S. A CPU é dividida em 3 seções: registradores de dados, unidade aritmético-lógica e unidade de controle.

Existem 16 registradores de dados de 16 bits cada, e com endereços hexadecimais 0,1,...,F, conhecidos como $R_0, R_1, ..., R_{15}$. A unidade de controle controla as operações da ULA e os acessos à memória e ao subsistema de E/S. Tem 2 registradores dedicados: o PC (*Program Counter*) que aponta para o endereço de memória que contém a próxima instrução e o RI (*registrador de instrução*) mantém a instrução que está sendo executada no ciclo corrente. Após cada ciclo, o PC é incrementado (quase sempre) em 1 e o RI é atualizado com a próxima instrução.

A memória principal tem 256 localizações de 16 bits cada. Seus endereços binários variam entre 00000000_2 e 11111101_2 ou entre 00_{16} e FD_{16}. A memória contém dados e instruções. As primeiras 64 localizações (de 00 a $3F_{16}$ são dedicadas a instruções que são armazenadas em posições consecutivas. Já a área de 40 a FD_{16} são utilizadas para armazenar dados.

Já o subsistema de E/S é bastante primitivo, consistindo de um teclado e um vídeo. Assume-se que o teclado usa a posição de memória FE_{16} e o monitor de vídeo, a posição FF_{16} se comportando como registradores de 16 bits interagindo com a CPU como uma posição de memória qualquer.

Conjunto de instruções Existem 16 instruções, mas só vamos usar 14. Cada instrução tem 2 partes o *opcode* e o(s) operando(s). O opcode indica o que fazer com o operando. Assim, a instrução (de 16 bits) está dividida em 4 campos de 4 bits cada. O primeiro sub-campo é o opcode e os outros 12 são interpretados de acordo com o opcode. Veja em



Veja que nem todas as operações exigem três operandos. O que não é usado não precisa estar preenchido com zeros. Um endereço de registrador é indicado por um único dígito hexadecimal e usa um único sub-campo, enquanto um endereço de memória usa dois dígitos hexadecimais (portanto 2 sub-campos). Veja as instruções projetadas

Instrução	Código	Operandos			Ação
	d_1	d_2	d_3	d_4	
HALT	0				Para a execução do programa
LOAD	1	R_D	M_F		$R_D \leftarrow M_F$
STORE	2	M_D		R_F	$M_D \leftarrow R_F$
ADDI	3	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
ADDF	4	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
MOVE	5	R_D	R_F		$R_D \leftarrow R_F$
NOT	6	R_D	R_F		$R_D \leftarrow \overline{R}_F$
AND	7	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ AND R_{F_2}
OR	8	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ OR R_{F_2}
XOR	9	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ XOR R_{F_2}
INC	A	R			$R \leftarrow R + 1$
DEC	B	R			$R \leftarrow R - 1$
ROTATE	C	R	n	0 ou 1	Rot_n R
JUMP	D	R	n		Se $R_0 \neq$ R então PC = *n*, do co

Legenda: R_F, R_{F_1}, R_{F_2}: endereço hexadecimal dos registradores fonte
R_D: endereço hexadecimal do registrador destino
M_F: endereço hexadecimal da localização de memória fonte
M_D: endereço hexadecimal da localização da memória destino
n: número hexadecimal
d_1, d_2, d_3, d_4: primeiro, segundo, terceiro e quarto dígitos hexadecimais

Exemplo Acompanhe a execução da adição de dois inteiros A e B colocando o resultado em C. Em alto nível a instrução é $C = A + B$. Para isso, o computador precisa colocar os 2 operandos em registradores distintos (por exemplo R_0 e R_1) e determinar que o resultado vá para outro registrador (por exemplo R_2). A ULA só pode operar se os dados estiverem em registradores. Mas, como há um número limitado de registradores, há um constante fluxo de vai-e-vem entre eles e a memória. Por simplicidade assume-se que os dois valores estão em na posição 40_{16} e 41_{16} enquanto o resultado será colocado na posição 42. Desta maneira esta soma precisa 5 instruções:

1. Carrega o conteúdo de M_{40} em R_0
2. carrega o conteúdo de M_{41} em R_1
3. Adiciona R_0 com R_1 e coloca o resultado em R_2
4. Armazena R_2 em M_{42}
5. termina o programa

Veja o programa em linguagem de máquina

Código	Interpretação			
$(1040)_{16}$	1: LOAD	0: R_0	40: M_{40}	
$(1141)_{16}$	1: LOAD	1: R_1	41: M_{41}	
$(3201)_{16}$	3: ADDI	2: R_2	0: R_0	1: R_1
$(2422)_{16}$	2: STORE	42: M_{42}		2: R_2
$(0000)_{16}$	0: HALT			

Perceba que lá na máquina o programa é `1040114132012422000`0 (é mole ?).

Note que isto é só uma pequena parte da coisa. Antes, precisaríamos ter digitado um valor, que obviamente entraria na posição do teclado e transferido este valor para o endereço 40. Depois a mesma coisa indo para o endereço 41, depois a execução deste programa e finalmente a transferência do valor do endereço 42 para o monitor de vídeo. Ufa! Apenas para sua referência o programa completo é

$$\underbrace{1FFE}_{1}\underbrace{240F}_{2}\underbrace{1FFE}_{3}\underbrace{241F}_{4}\underbrace{1040}_{5}\underbrace{1141}_{6}\underbrace{3201}_{7}\underbrace{2422}_{8}$$

$$\underbrace{1F42}_{9}\underbrace{2FFF}_{10}\underbrace{0000}_{11}$$

As instruções 1 a 4 são para entrada e as 9 a 10 para saída.

Exemplo Nos exercicios a seguir, considere que os dados ja foram lidos nas posições de memória citadas e que serão exibidos a partir das posições também citadas. O que você deve resolver é a tradução das expressões aritméticas para linguagem de máquina e vice versa. Para facilitar a correção, use sempre o menor registrador disponível no momento.

Pede-se que você traduza para a linguagem de máquina a expressão $C \leftarrow A + (2 * B)$, sendo que &A=61, &B=65 e &D=72. Eis como ficaria

```
1065 - LOAD,0,65
3100 - ADDI,1,0,0  // 1=0+0
1261 - LOAD,2,61
3312 - ADDI,3,1,2  // 3=1+2
2723 - STORE,72,3
em linguagem de máquina 1065.3100.1261.3312.2723
```

Pede-se também o que faz o programa

`1060.3100.3210.B200.B211.2732`

A instrução `1060`, carrega no R_0 o conteúdo do endereço 60.
A instrução `3100`, soma os conteudos de R_0 e R_0 colocando o resultado em R_1, ou $R_1 = R_0 + R_0$
A instrução `3210`, soma os conteudos de R_1 e R_0 colocando o resultado em R_2, ou $R_2 = R_1 + R_0$ que devidamente interpretado com o anterior faz $R_2 = 3 \times R_0$
Depois, a instrução `B200`, decrementa o R_2. Note que o `00` que termina a instrução é puro lixo e não interessa. Outra maneira de representar esta instrução é `R2??` Mais um `B211`, que novamente decrementa o R_2. Agora o lixo é `11`
Finalmente, `2732` armazena o R_2 no endereço 73. Em resumo, aquele programa calcula $(X)_{73} \leftarrow ((Y)_{60} \times 3) - 2$

☞ Para você fazer

1. Traduza para linguagem de máquina a expressão $C = B - 1$,com &C=45,&B=60
2. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1060.A0??.A0??.2450
3. Traduza para linguagem de máquina a expressão $D = A + B + C$,com &A=40,&B=50,&C=60,&D=70
4. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.3100.3101.2701

1

2

3

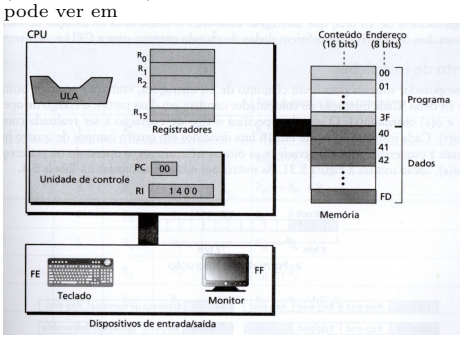
4

205-76625 - n lim

CEP - UP - UTFPR - PUCPr - UFPR –
10/06/2024 - 08:33:04.1
Prof Dr P Kantek (pkantek@gmail.com)
Um computador simples VIVXq06a, V: 1.31
76632 VICTOR BRONDANI DOS SANTOS
24CC2205 - 15 27/jun limite 1. avaliacao _____ /
/ _____

Um computador simples

Vamos projetar um computador simples, para estudar de perto a sua arquitetura e seu processamento de instruções. O fato de ser um computador teórico (portanto irreal) não atrapalha. Projetado como se pode ver em



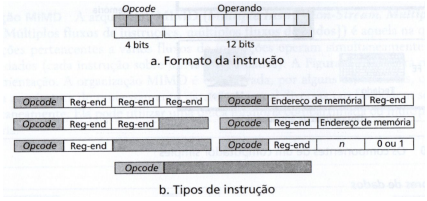
o computador terá 3 componentes: CPU, memória e subsistema de E/S. A CPU é dividida em 3 seções: registradores de dados, unidade aritmético-lógica e unidade de controle.

Existem 16 registradores de dados de 16 bits cada, e com endereços hexadecimais 0,1,...,F, conhecidos como $R_0, R_1, ..., R_{15}$. A unidade de controle controla as operações da ULA e os acessos à memória e ao subsistema de E/S. Tem 2 registradores dedicados: o PC (*Program Counter*) que aponta para o endereço de memória que contém a próxima instrução e o RI (*registrador de instrução*) mantém a instrução que está sendo executada no ciclo corrente. Após cada ciclo, o PC é incrementado (quase sempre) em 1 e o RI é atualizado com a próxima instrução.

A memória principal tem 256 localizações de 16 bits cada. Seus endereços binários variam entre 00000000_2 e 11111101_2 ou entre 00_{16} e FD_{16}. A memória contém dados e instruções. As primeiras 64 localizações (de 00 a $3F_{16}$ são dedicadas a instruções que são armazenadas em posições consecutivas. Já a área de 40 a FD_{16} são utilizadas para armazenar dados.

Já o subsistema de E/S é bastante primitivo, consistindo de um teclado e um vídeo. Assume-se que o teclado usa a posição de memória FE_{16} e o monitor de vídeo, a posição FF_{16} se comportando como registradores de 16 bits interagindo com a CPU como uma posição de memória qualquer.

Conjunto de instruções Existem 16 instruções, mas só vamos usar 14. Cada instrução tem 2 partes o *opcode* e o(s) operando(s). O opcode indica o que fazer com o operando. Assim, a instrução (de 16 bits) está dividida em 4 campos de 4 bits cada. O primeiro sub-campo é o opcode e os outros 12 são interpretados de acordo com o opcode. Veja em



Veja que nem todas as operações exigem três operandos. O que não é usado não precisa estar preenchido com zeros. Um endereço de registrador é indicado por um único dígito hexadecimal e usa um único sub-campo, enquanto um endereço de memória usa dois dígitos hexadecimais (portanto 2 sub-campos). Veja as instruções projetadas

Instrução	Código	Operandos			Ação
	d_1	d_2	d_3	d_4	
HALT	0				Para a execução do programa
LOAD	1	R_D	M_F		$R_D \leftarrow M_F$
STORE	2	M_D		R_F	$M_D \leftarrow R_F$
ADDI	3	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
ADDF	4	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
MOVE	5	R_D	R_F		$R_D \leftarrow R_F$
NOT	6	R_D	R_F		$R_D \leftarrow \overline{R}_F$
AND	7	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ AND R_{F_2}
OR	8	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ OR R_{F_2}
XOR	9	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ XOR R_{F_2}
INC	A	R			$R \leftarrow R + 1$
DEC	B	R			$R \leftarrow R - 1$
ROTATE	C	R	n	0 ou 1	Rot_n R
JUMP	D	R	n		Se $R_0 \neq$ R então PC = *n*, do co

Legenda: R_F, R_{F_1}, R_{F_2}: endereço hexadecimal dos registradores fonte
R_D: endereço hexadecimal do registrador destino
M_F: endereço hexadecimal da localização de memória fonte
M_D: endereço hexadecimal da localização da memória destino
n: número hexadecimal
d_1, d_2, d_3, d_4: primeiro, segundo, terceiro e quarto dígitos hexadecimais

Exemplo Acompanhe a execução da adição de dois inteiros A e B colocando o resultado em C. Em alto nível a instrução é $C = A + B$. Para isso, o computador precisa colocar os 2 operandos em registradores distintos (por exemplo R_0 e R_1) e determinar que o resultado vá para outro registrador (por exemplo R_2). A ULA só pode operar se os dados estiverem em registradores. Mas, como há um número limitado de registradores, há um constante fluxo de vai-e-vem entre eles e a memória. Por simplicidade assume-se que os dois valores estão em na posição 40_{16} e 41_{16} enquanto o resultado será colocado na posição 42. Desta maneira esta soma precisa 5 instruções:

1. Carrega o conteúdo de M_{40} em R_0
2. carrega o conteúdo de M_{41} em R_1
3. Adiciona R_0 com R_1 e coloca o resultado em R_2
4. Armazena R_2 em M_{42}
5. termina o programa

Veja o programa em linguagem de máquina

Código	Interpretação			
$(1040)_{16}$	1: LOAD	0: R_0	40: M_{40}	
$(1141)_{16}$	1: LOAD	1: R_1	41: M_{41}	
$(3201)_{16}$	3: ADDI	2: R_2	0: R_0	1: R_1
$(2422)_{16}$	2: STORE	42: M_{42}		2: R_2
$(0000)_{16}$	0: HALT			

Perceba que lá na máquina o programa é `1040114132012422 0000` (é mole ?).

Note que isto é só uma pequena parte da coisa. Antes, precisaríamos ter digitado um valor, que obviamente entraria na posição do teclado e transferido este valor para o endereço 40. Depois a mesma coisa indo para o endereço 41, depois a execução deste programa e finalmente a transferência do valor do endereço 42 para o monitor de vídeo. Ufa! Apenas para sua referência o programa completo é

$$\underbrace{1FFE}_{1}\ \underbrace{240F}_{2}\ \underbrace{1FFE}_{3}\ \underbrace{241F}_{4}\ \underbrace{1040}_{5}\ \underbrace{1141}_{6}\ \underbrace{3201}_{7}\ \underbrace{2422}_{8}$$

$$\underbrace{1F42}_{9}\ \underbrace{2FFF}_{10}\ \underbrace{0000}_{11}$$

As instruções 1 a 4 são para entrada e as 9 a 10 para saída.

Exemplo Nos exercicios a seguir, considere que os dados ja foram lidos nas posições de memória citadas e que serão exibidos a partir das posições também citadas. O que você deve resolver é a tradução das expressões aritméticas para linguagem de máquina e vice versa. Para facilitar a correção, use sempre o menor registrador disponível no momento.

Pede-se que você traduza para a linguagem de máquina a expressão $C \leftarrow A + (2 * B)$, sendo que &A=61, &B=65 e &D=72. Eis como ficaria

```
1065 - LOAD,0,65
3100 - ADDI,1,0,0  // 1=0+0
1261 - LOAD,2,61
3312 - ADDI,3,1,2  // 3=1+2
2723 - STORE,72,3
em linguagem de máquina 1065.3100.1261.3312.2723
```

Pede-se também o que faz o programa

```
1060.3100.3210.B200.B211.2732
```

A instrução `1060`, carrega no R_0 o conteúdo do endereço 60.
A instrução `3100`, soma os conteudos de R_0 e R_0 colocando o resultado em R_1, ou $R_1 = R_0 + R_0$
A instrução `3210`, soma os conteudos de R_1 e R_0 colocando o resultado em R_2, ou $R_2 = R_1 + R_0$ que devidamente interpretado com o anterior faz $R_2 = 3 \times R_0$
Depois, a instrução `B200`, decrementa o R_2. Note que o `00` que termina a instrução é puro lixo e não interessa. Outra maneira de representar esta instrução é `R2??` Mais um `B211`, que novamente decrementa o R_2. Agora o lixo é `11`
Finalmente, `2732` armazena o R_2 no endereço 73. Em resumo, aquele programa calcula $(X)_{73} \leftarrow ((Y)_{60} \times 3) - 2$

☞ Para você fazer

1. Traduza para linguagem de máquina a expressão $D = A + A + A + A + A$, &A=45,&D=60
2. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.3100.2701
3. Traduza para linguagem de máquina a expressão $B = A + 3$,com &A=45,&B=60
4. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.A0??.2700

1

2

3

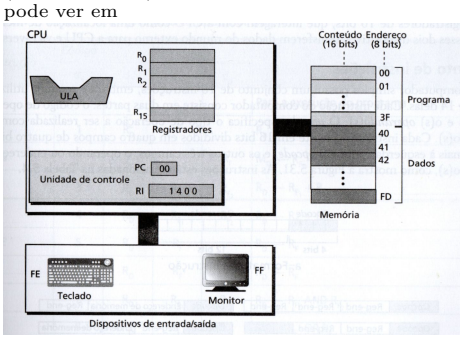
4

205-76632 - n lim

CEP - UP - UTFPR - PUCPr - UFPR –
10/06/2024 - 08:33:04.1
Prof Dr P Kantek (pkantek@gmail.com)
Um computador simples VIVXq06a, V: 1.31
76649 VICTOR HUGO DOS SANTOS DE
CAMA 24CC2205 - 16 27/jun limite 1. avaliacao
/ _____ / _____

Um computador simples

Vamos projetar um computador simples, para estudar de perto a sua arquitetura e seu processamento de instruções. O fato de ser um computador teórico (portanto irreal) não atrapalha. Projetado como se pode ver em



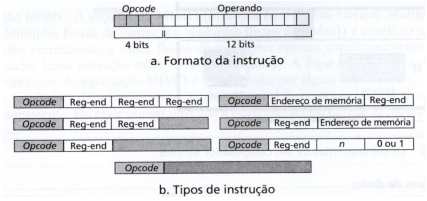
o computador terá 3 componentes: CPU, memória e subsistema de E/S. A CPU é dividida em 3 seções: registradores de dados, unidade aritmético-lógica e unidade de controle.

Existem 16 registradores de dados de 16 bits cada, e com endereços hexadecimais 0,1,...,F, conhecidos como $R_0, R_1, ..., R_{15}$. A unidade de controle controla as operações da ULA e os acessos à memória e ao subsistema de E/S. Tem 2 registradores dedicados: o PC (*Program Counter*) que aponta para o endereço de memória que contém a próxima instrução e o RI (*registrador de instrução*) mantém a instrução que está sendo executada no ciclo corrente. Após cada ciclo, o PC é incrementado (quase sempre) em 1 e o RI é atualizado com a próxima instrução.

A memória principal tem 256 localizações de 16 bits cada. Seus endereços binários variam entre 00000000_2 e 11111101_2 ou entre 00_{16} e FD_{16}. A memória contém dados e instruções. As primeiras 64 localizações (de 00 a $3F_{16}$ são dedicadas a instruções que são armazenadas em posições consecutivas. Já a área de 40 a FD_{16} são utilizadas para armazenar dados.

Já o subsistema de E/S é bastante primitivo, consistindo de um teclado e um vídeo. Assume-se que o teclado usa a posição de memória FE_{16} e o monitor de vídeo, a posição FF_{16} se comportando como registradores de 16 bits interagindo com a CPU como uma posição de memória qualquer.

Conjunto de instruções Existem 16 instruções, mas só vamos usar 14. Cada instrução tem 2 partes o *opcode* e o(s) operando(s). O opcode indica o que fazer com o operando. Assim, a instrução (de 16 bits) está dividida em 4 campos de 4 bits cada. O primeiro sub-campo é o opcode e os outros 12 são interpretados de acordo com o opcode. Veja em



Veja que nem todas as operações exigem três operandos. O que não é usado não precisa estar preenchido com zeros. Um endereço de registrador é indicado por um único dígito hexadecimal e usa um único sub-campo, enquanto um endereço de memória usa dois dígitos hexadecimais (portanto 2 sub-campos). Veja as instruções projetadas

Instrução	Código	Operandos				Ação
	d_1	d_2	d_3	d_4		
HALT	0					Para a execução do programa
LOAD	1	R_D	M_F			$R_D \leftarrow M_F$
STORE	2	M_D		R_F		$M_D \leftarrow R_F$
ADDI	3	R_D	R_{F_1}	R_{F_2}		$R_D \leftarrow R_{F_1} + R_{F_2}$
ADDF	4	R_D	R_{F_1}	R_{F_2}		$R_D \leftarrow R_{F_1} + R_{F_2}$
MOVE	5	R_D	R_F			$R_D \leftarrow R_F$
NOT	6	R_D	R_F			$R_D \leftarrow \overline{R}_F$
AND	7	R_D	R_{F_1}	R_{F_2}		$R_D \leftarrow R_{F_1}$ AND R_{F_2}
OR	8	R_D	R_{F_1}	R_{F_2}		$R_D \leftarrow R_{F_1}$ OR R_{F_2}
XOR	9	R_D	R_{F_1}	R_{F_2}		$R_D \leftarrow R_{F_1}$ XOR R_{F_2}
INC	A	R				$R \leftarrow R + 1$
DEC	B	R				$R \leftarrow R - 1$
ROTATE	C	R	n	0 ou 1		Rot_n R
JUMP	D	R	n			Se $R_0 \neq$ R então PC = *n*, do co

Legenda: R_F, R_{F_1}, R_{F_2}: endereço hexadecimal dos registradores fonte
R_D: endereço hexadecimal do registrador destino
M_F: endereço hexadecimal da localização de memória fonte
M_D: endereço hexadecimal da localização da memória destino
n: número hexadecimal
d_1, d_2, d_3, d_4: primeiro, segundo, terceiro e quarto dígitos hexadecimais

Exemplo Acompanhe a execução da adição de dois inteiros A e B colocando o resultado em C. Em alto nível a instrução é $C = A + B$. Para isso, o computador precisa colocar os 2 operandos em registradores distintos (por exemplo R_0 e R_1) e determinar que o resultado vá para outro registrador (por exemplo R_2). A ULA só pode operar se os dados estiverem em registradores. Mas, como há um número limitado de registradores, há um constante fluxo de vai-e-vem entre eles e a memória. Por simplicidade assume-se que os dois valores estão em na posição 40_{16} e 41_{16} enquanto o resultado será colocado na posição 42. Desta maneira esta soma precisa 5 instruções:

1. Carrega o conteúdo de M_{40} em R_0
2. carrega o conteúdo de M_{41} em R_1
3. Adiciona R_0 com R_1 e coloca o resultado em R_2
4. Armazena R_2 em M_{42}
5. termina o programa

Veja o programa em linguagem de máquina

Código	Interpretação			
$(1040)_{16}$	1: LOAD	0: R_0	40: M_{40}	
$(1141)_{16}$	1: LOAD	1: R_1	41: M_{41}	
$(3201)_{16}$	3: ADDI	2: R_2	0: R_0	1: R_1
$(2422)_{16}$	2: STORE	42: M_{42}		2: R_2
$(0000)_{16}$	0: HALT			

Perceba que lá na máquina o programa é `1040114132012422 0000` (é mole ?).

Note que isto é só uma pequena parte da coisa. Antes, precisaríamos ter digitado um valor, que obviamente entraria na posição do teclado e transferido este valor para o endereço 40. Depois a mesma coisa indo para o endereço 41, depois a execução deste programa e finalmente a transferência do valor do endereço 42 para o monitor de vídeo. Ufa! Apenas para sua referência o programa completo é

$$\underbrace{1FFE}_{1}\underbrace{240F}_{2}\underbrace{1FFE}_{3}\underbrace{241F}_{4}\underbrace{1040}_{5}\underbrace{1141}_{6}\underbrace{3201}_{7}\underbrace{2422}_{8}$$

$$\underbrace{1F42}_{9}\underbrace{2FFF}_{10}\underbrace{0000}_{11}$$

As instruções 1 a 4 são para entrada e as 9 a 10 para saída.

Exemplo Nos exercicios a seguir, considere que os dados ja foram lidos nas posições de memória citadas e que serão exibidos a partir das posições também citadas. O que você deve resolver é a tradução das expressões aritméticas para linguagem de máquina e vice versa. Para facilitar a correção, use sempre o menor registrador disponível no momento.

Pede-se que você traduza para a linguagem de máquina a expressão $C \leftarrow A + (2 * B)$, sendo que &A=61, &B=65 e &D=72. Eis como ficaria

```
1065 - LOAD,0,65
3100 - ADDI,1,0,0  // 1=0+0
1261 - LOAD,2,61
3312 - ADDI,3,1,2  // 3=1+2
2723 - STORE,72,3
em linguagem de máquina 1065.3100.1261.3312.2723
```

Pede-se também o que faz o programa

```
1060.3100.3210.B200.B211.2732
```

A instrução `1060`, carrega no R_0 o conteúdo do endereço 60.
A instrução `3100`, soma os conteudos de R_0 e R_0 colocando o resultado em R_1, ou $R_1 = R_0 + R_0$
A instrução `3210`, soma os conteudos de R_1 e R_0 colocando o resultado em R_2, ou $R_2 = R_1 + R_0$ que devidamente interpretado com o anterior faz $R_2 = 3 \times R_0$
Depois, a instrução `B200`, decrementa o R_2. Note que o `00` que termina a instrução é puro lixo e não interessa. Outra maneira de representar esta instrução é `R2??` Mais um `B211`, que novamente decrementa o R_2. Agora o lixo é `11`
Finalmente, `2732` armazena o R_2 no endereço 73. Em resumo, aquele programa calcula $(X)_{73} \leftarrow ((Y)_{60} \times 3) - 2$

☞ Para você fazer

1. Traduza para linguagem de máquina a expressão $B = A + 3$,com &A=45,&B=60
2. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.3100.3101.3101.3101.2601
3. Traduza para linguagem de máquina a expressão $B = A - 2$,com &A=45,&B=60
4. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1060.B0??.2450

1

2

3

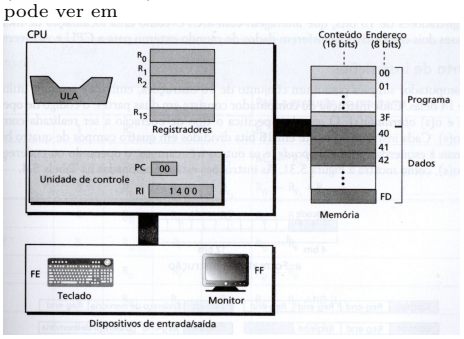
4

205-76649 - n lim

CEP - UP - UTFPR - PUCPr - UFPR –
10/06/2024 - 08:33:04.1
Prof Dr P Kantek (pkantek@gmail.com)
Um computador simples VIVXq06a, V: 1.31
76656 VITOR EDUARDO DE GOES FONTES
24CC2205 - 17 27/jun limite 1. avaliacao _____ /
/ _____

Um computador simples

Vamos projetar um computador simples, para estudar de perto a sua arquitetura e seu processamento de instruções. O fato de ser um computador teórico (portanto irreal) não atrapalha. Projetado como se pode ver em



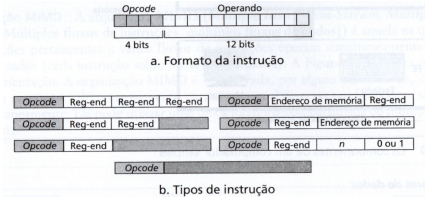
o computador terá 3 componentes: CPU, memória e subsistema de E/S. A CPU é dividida em 3 seções: registradores de dados, unidade aritmético-lógica e unidade de controle.

Existem 16 registradores de dados de 16 bits cada, e com endereços hexadecimais 0,1,...,F, conhecidos como $R_0, R_1, ..., R_{15}$. A unidade de controle controla as operações da ULA e os acessos à memória e ao subsistema de E/S. Tem 2 registradores dedicados: o PC (*Program Counter*) que aponta para o endereço de memória que contém a próxima instrução e o RI (*registrador de instrução*) mantém a instrução que está sendo executada no ciclo corrente. Após cada ciclo, o PC é incrementado (quase sempre) em 1 e o RI é atualizado com a próxima instrução.

A memória principal tem 256 localizações de 16 bits cada. Seus endereços binários variam entre 00000000_2 e 11111101_2 ou entre 00_{16} e FD_{16}. A memória contém dados e instruções. As primeiras 64 localizações (de 00 a $3F_{16}$ são dedicadas a instruções que são armazenadas em posições consecutivas. Já a área de 40 a FD_{16} são utilizadas para armazenar dados.

Já o subsistema de E/S é bastante primitivo, consistindo de um teclado e um vídeo. Assume-se que o teclado usa a posição de memória FE_{16} e o monitor de vídeo, a posição FF_{16} se comportando como registradores de 16 bits interagindo com a CPU como uma posição de memória qualquer.

Conjunto de instruções Existem 16 instruções, mas só vamos usar 14. Cada instrução tem 2 partes o *opcode* e o(s) operando(s). O opcode indica o que fazer com o operando. Assim, a instrução (de 16 bits) está dividida em 4 campos de 4 bits cada. O primeiro sub-campo é o opcode e os outros 12 são interpretados de acordo com o opcode. Veja em



Veja que nem todas as operações exigem três operandos. O que não é usado não precisa estar preenchido com zeros. Um endereço de registrador é indicado por um único dígito hexadecimal e usa um único sub-campo, enquanto um endereço de memória usa dois dígitos hexadecimais (portanto 2 sub-campos). Veja as instruções projetadas

Instrução	Código	Operandos			Ação
	d_1	d_2	d_3	d_4	
HALT	0				Para a execução do programa
LOAD	1	R_D	M_F		$R_D \leftarrow M_F$
STORE	2	M_D		R_F	$M_D \leftarrow R_F$
ADDI	3	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
ADDF	4	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1} + R_{F_2}$
MOVE	5	R_D	R_F		$R_D \leftarrow R_F$
NOT	6	R_D	R_F		$R_D \leftarrow \overline{R}_F$
AND	7	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ AND R_{F_2}
OR	8	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ OR R_{F_2}
XOR	9	R_D	R_{F_1}	R_{F_2}	$R_D \leftarrow R_{F_1}$ XOR R_{F_2}
INC	A	R			$R \leftarrow R + 1$
DEC	B	R			$R \leftarrow R - 1$
ROTATE	C	R	n	0 ou 1	Rot_n R
JUMP	D	R	n		Se $R_0 \neq$ R então PC = *n*, do co

Legenda: R_F, R_{F_1}, R_{F_2}: endereço hexadecimal dos registradores fonte
R_D: endereço hexadecimal do registrador destino
M_F: endereço hexadecimal da localização de memória fonte
M_D: endereço hexadecimal da localização da memória destino
n: número hexadecimal
d_1, d_2, d_3, d_4: primeiro, segundo, terceiro e quarto dígitos hexadecimais

Exemplo Acompanhe a execução da adição de dois inteiros A e B colocando o resultado em C. Em alto nível a instrução é $C = A + B$. Para isso, o computador precisa colocar os 2 operandos em registradores distintos (por exemplo R_0 e R_1) e determinar que o resultado vá para outro registrador (por exemplo R_2). A ULA só pode operar se os dados estiverem em registradores. Mas, como há um número limitado de registradores, há um constante fluxo de vai-e-vem entre eles e a memória. Por simplicidade assume-se que os dois valores estão em na posição 40_{16} e 41_{16} enquanto o resultado será colocado na posição 42. Desta maneira esta soma precisa 5 instruções:

1. Carrega o conteúdo de M_{40} em R_0
2. carrega o conteúdo de M_{41} em R_1
3. Adiciona R_0 com R_1 e coloca o resultado em R_2
4. Armazena R_2 em M_{42}
5. termina o programa

Veja o programa em linguagem de máquina

Código	Interpretação			
$(1040)_{16}$	1: LOAD	0: R_0	40: M_{40}	
$(1141)_{16}$	1: LOAD	1: R_1	41: M_{41}	
$(3201)_{16}$	3: ADDI	2: R_2	0: R_0	1: R_1
$(2422)_{16}$	2: STORE	42: M_{42}		2: R_2
$(0000)_{16}$	0: HALT			

Perceba que lá na máquina o programa é `1040114132012422 0000` (é mole ?).

Note que isto é só uma pequena parte da coisa. Antes, precisaríamos ter digitado um valor, que obviamente entraria na posição do teclado e transferido este valor para o endereço 40. Depois a mesma coisa indo para o endereço 41, depois a execução deste programa e finalmente a transferência do valor do endereço 42 para o monitor de vídeo. Ufa! Apenas para sua referência o programa completo é

$$\underbrace{1FFE}_{1}\ \underbrace{240F}_{2}\ \underbrace{1FFE}_{3}\ \underbrace{241F}_{4}\ \underbrace{1040}_{5}\ \underbrace{1141}_{6}\ \underbrace{3201}_{7}\ \underbrace{2422}_{8}$$

$$\underbrace{1F42}_{9}\ \underbrace{2FFF}_{10}\ \underbrace{0000}_{11}$$

As instruções 1 a 4 são para entrada e as 9 a 10 para saída.

Exemplo Nos exercicios a seguir, considere que os dados ja foram lidos nas posições de memória citadas e que serão exibidos a partir das posições também citadas. O que você deve resolver é a tradução das expressões aritméticas para linguagem de máquina e vice versa. Para facilitar a correção, use sempre o menor registrador disponível no momento.

Pede-se que você traduza para a linguagem de máquina a expressão $C \leftarrow A + (2 * B)$, sendo que &A=61, &B=65 e &D=72. Eis como ficaria

```
1065 - LOAD,0,65
3100 - ADDI,1,0,0  // 1=0+0
1261 - LOAD,2,61
3312 - ADDI,3,1,2  // 3=1+2
2723 - STORE,72,3
em linguagem de máquina 1065.3100.1261.3312.2723
```

Pede-se também o que faz o programa

```
1060.3100.3210.B200.B211.2732
```

A instrução `1060`, carrega no R_0 o conteúdo do endereço 60.
A instrução `3100`, soma os conteudos de R_0 e R_0 colocando o resultado em R_1, ou $R_1 = R_0 + R_0$
A instrução `3210`, soma os conteudos de R_1 e R_0 colocando o resultado em R_2, ou $R_2 = R_1 + R_0$ que devidamente interpretado com o anterior faz $R_2 = 3 \times R_0$
Depois, a instrução `B200`, decrementa o R_2. Note que o `00` que termina a instrução é puro lixo e não interessa. Outra maneira de representar esta instrução é `R2??` Mais um `B211`, que novamente decrementa o R_2. Agora o lixo é `11`
Finalmente, `2732` armazena o R_2 no endereço 73. Em resumo, aquele programa calcula $(X)_{73} \leftarrow ((Y)_{60} \times 3) - 2$

☞ Para você fazer

1. Traduza para linguagem de máquina a expressão $B = A - 2$,com &A=45,&B=60
2. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.A0??.A0??.A0??.2600
3. Traduza para linguagem de máquina a expressão $D = A + A + A + A + A$, &A=45,&D=60
4. Descreva em linguagem matemática o que faz o programa em linguagem de máquina 1045.3100.3101.2701

1

2

3

4

205-76656 - n lim