Arquitetura e organização de computadores

Representação de instruções

Representação de instruções

- **INSTRUÇÃO DE MÁQUINA**

- Quem executa um programa é o hardware e o que ele espera encontrar é um programa em linguagem de máquina (uma seqüência de instruções de máquina em código binário).

- Um programa em linguagem de alto nível não pode ser executado diretamente pelo *hardware*.

Representação de instruções

- **INSTRUÇÃO DE MÁQUINA**

- O programa tem que ser transformado (traduzido) para linguagem de máquina por um compilador, antes de ser carregada em memória.
- Para que o hardware possa executá-lo. A linguagem de máquina é composta de códigos binários, representando instruções, endereços e dados e está totalmente vinculada ao conjunto (*"set"*) de instruções da máquina.

Representação de instruções

Funcionalmente as operações do computador são:

- **matemáticas** (aritméticas, lógicas, de complemento, de deslocamento...
- **movimentação de dados** (memória <--> registrador)
- **entrada-saída** (leitura e escrita em dispositivos externos - dispositivos de Entrada / Saída)
- **controle** (desvio da sequência de execução, parar, etc...)

Representação de instruções

Exemplo:

O conjunto de instruções de alguns processadores, como por exemplo o Intel 8080, não possui instruções para multiplicação ou divisão; portanto, um programa em linguagem de máquina, nestes processadores, não pode fazer multiplicação ou divisão diretamente (somente através de combinação de outras instruções).

Já um programa em linguagem de nível mais alto pode implementar comandos de multiplicação (ou divisão) que combinem uma série de instruções binárias do conjunto de instruções para fazer uma multiplicação (ou divisão) através de repetidas somas (subtrações) ou por deslocamento de bits.

Representação de instruções

Exemplo: Conjunto de instruções
Máquina hipotética

Instrução	Significado	Operação	Código
Load	Carregar no acumulador	ACC <-- op	0000
Store	Salvar na memória	op <-- ACC	0001
Add	Somar	ACC <-- ACC + op	0010
Sub	Subtrair	ACC <-- ACC - op	0011
Mult	Multiplicar	ACC <-- ACC * op	0100
Div	Dividir	ACC <-- ACC / op	0101
Jmp	Desviar	CI <-- op	0110
Jz	Desviar, se ACC igual zero	CI <-- op, se ACC = 0	0111
Jnz	Desviar, se ACC não zero	CI <-- op, se ACC != 0	1000
Read	Ler entrada	op <-- entrada	1001
Print	Imprimir	saida <-- op	1010
Stop	Terminar		1100

Representação de instruções

Convenções usadas:

Operador	Significado	Operador	Significado
+	adição	~	not (negação)
-	subtração	=	igual
*	multiplicação	!=	diferente
/	divisão	|	or (AND)
<--	atribuição	^	exclusive or (XOR)
<<	deslocamento à esquerda	&	and (E)
>>	deslocamento à direita		

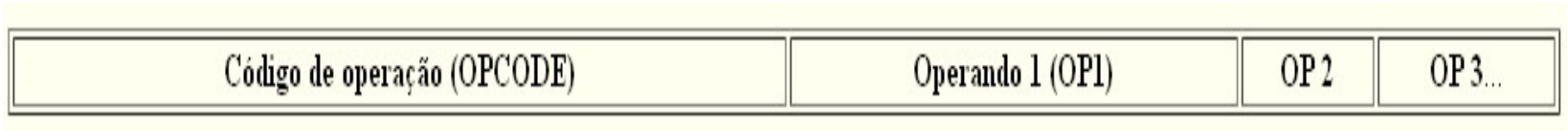
Representação de instruções

Formato das instruções:

- **Código de Operação ou OPCODE** - identifica a operação a ser realizada pelo processador. É o campo da instrução cuja valor binário identifica (é o código binário) da operação a ser realizada. Este código é a entrada no decodificador de instruções na unidade de controle.
- Cada instrução deverá ter um código único que a identifique.

Representação de instruções

Formato das instruções:



- **Operando(s)** - é o campo da instrução cujo valor binário sinaliza a localização do dado (ou é o próprio dado) que será manipulado (processado) pela instrução durante a operação.

Representação de instruções

Formato das instruções:

- Em geral, um operando identifica o endereço de memória onde está contido o dado que será manipulado, ou pode conter o endereço onde o resultado da operação será armazenado. Finalmente, um operando pode também indicar um Registrador (que conterá o dado propriamente dito ou um endereço de memória onde está armazenado o dado). Os operandos na verdade fornecem os dados da instrução. Obs: Existem instruções que não tem operando. Ex.: Instrução HALT (PARE).

Representação de instruções

Formatos de instrução:

Há diversos formatos de instruções, com características particulares, vantagens e desvantagens.
O conjunto de instruções de uma máquina pode ser constituído por instruções de diversos formatos. Esta flexibilidade permite a escolha da instrução adequada para aplicação em cada caso.
Conjunto de instruções pode ser analisado sob alguns aspectos, por exemplo:

- quantidade de instruções
- quantidade de operandos
- modo de endereçamento (próxima aula)

Representação de instruções

Formatos de instrução:

Exemplos

ADD OP1 OP2	=>	(OP1) <-- (OP1) + (OP2)	Soma conteúdo de OP1 e OP2 e armazena o resultado em OP1
ADD OP1 OP2 OP3	=>	(OP3) <-- (OP1) + (OP2)	Soma conteúdo de OP1 e OP2 e armazena o resultado em OP3
ADD OP1	=>	(ACC)<--(ACC) + (OP1)	Soma conteúdo de OP1 e ACC e armazena o resultado em ACC

Representação de instruções

Formatos de instrução:

LTR - Linguagem de Transferência entre Registradores (register transfer language)

A LTR é utilizada para sinalizar o processamento de uma operação, mostrando o que acontece com os registradores e posições de memória como resultado do processamento de uma instrução.

- caracteres alfanuméricos significam abreviaturas de nomes de registradores ou posições de memória (ex: REM. MP)
- parênteses indicam conteúdo, no caso de registradores, ou o valor é um endereço na MP.
- seta indica atribuição (transferência de conteúdo entre registradores ou entre registrador e memória principal)

Representação de instruções

Formatos de instrução:

LTR - Linguagem de Transferência entre Registradores

Exemplo:

- REM <--- (CI)

  o conteúdo do CI é copiado para o REM

- RDM <--- (MP(REM))

  o conteúdo da célula da MP cujo endereço está no REM é copiado para o RDM

Representação de instruções

TAMANHO (EM BITS) DE UMA INSTRUÇÃO

PROBLEMA DE PROJETO: Escolha do tamanho das instruções

Da escolha do tamanho das instruções depende :

- tamanho da memória
- tamanho das células da MP
- velocidade de acesso
- organização do barramento de dados.

Representação de instruções

TAMANHO (EM BITS) DE UMA INSTRUÇÃO

Basicamente, o projetista decide entre fatores como economia de espaço em memória, processamento mais rápido das instruções ou um conjunto de instruções mais completo e poderoso:

Maior Conjunto Instruções

--> requer muitas instruções (para atendimento a diferentes aplicações) --->muitos bits por *opcode*
--> requer instruções completas ---> muitos bits para operandos
--> requer *hardware* mais complexo ---> processamento de instruções mais lento

Representação de instruções

OPERANDO

Como geralmente o operando contém um endereço da MP, o número de bits ocupado por um operando depende do número de células endereçáveis de memória, ou seja, é preciso saber quantas células podem ser endereçadas na memória principal para saber quantos bits serão necessários para o operando.
Obs.: desta forma, pode-se concluir que o número de bits de cada operando será igual ao número de bits do REM e do CI.

Representação de instruções

OPERANDO

Por simplicidade vamos **considerar que todas as instruções de uma máquina tem o mesmo tamanho**. Esta simplificação muitas vezes não corresponde à realidade de uma máquina comercial, mas atende aos objetivos deste curso.

Representação de instruções

Exercícios:

Exercício 3:

Comente quais as conseqüências se, no problema anterior, fosse alterado o item a) para 8 instruções.

Operações de Instruções

Representação de instruções

- Conceitos;
- Quantidade de operandos;
- Instruções com 3 operandos;
- Instruções com 2 operandos;
- Instruções com 1 operando;
- Modos de endereçamento.

Conceitos

- Quem executa um programa é o hardware e o que ele espera encontrar é um programa em linguagem de máquina (uma sequência de instruções de máquina em código binário). Um programa em linguagem de alto nível não pode ser executado diretamente pelo *hardware*. Ele tem que ser transformado (traduzido) para linguagem de máquina por um compilador, antes de ser carregada em memória, para que o hardware possa executá-lo. A linguagem de máquina é composta de códigos binários, representando instruções, endereços e dados e está totalmente vinculada ao conjunto (*"set"*) de instruções da máquina.

- Funcionalmente as operações do computador são:
  · matemáticas (aritméticas, lógicas, de complemento, de deslocamento...)
  · movimentação de dados (memória <--> registrador)
  · entrada-saída (leitura e escrita em dispositivos externos - dispositivos de Entrada / Saída)
  · controle (desvio da sequência de execução, parar, etc...)

- **FORMATO DAS INSTRUÇÕES**

Código OP	Operando 1	Operando 2	Operando 3

- **Código de Operação ou OPCODE** - identifica a operação a ser realizada pelo processador. É o campo da instrução cuja valor binário identifica (é o código binário) da operação a ser realizada. Cada instrução deverá ter um código único que a identifique.

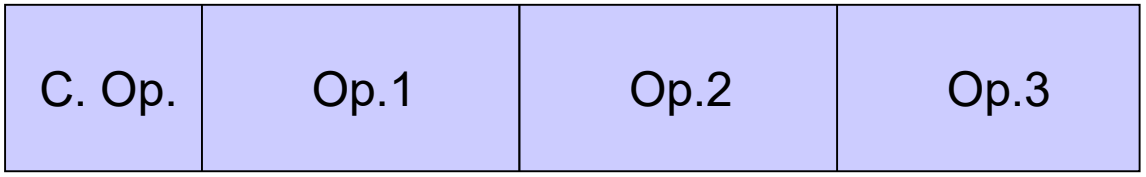
- **Operando(s)** - é o campo da instrução cujo valor binário sinaliza a localização do dado (ou é o próprio dado) que será manipulado (processado) pela instrução durante a operação.

BASICAMENTE

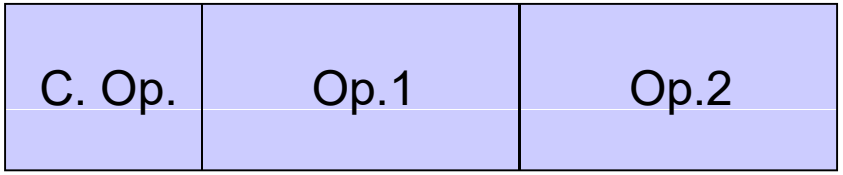
- C. Op. – indica ao processador o que fazer e como fazer.
- Op. – indica ao processador com que dado ou dados a operação irá se realizar.

Esses elementos são identificados para o processador por um grupo de bits específicos, os quais em conjunto, formam a instrução.

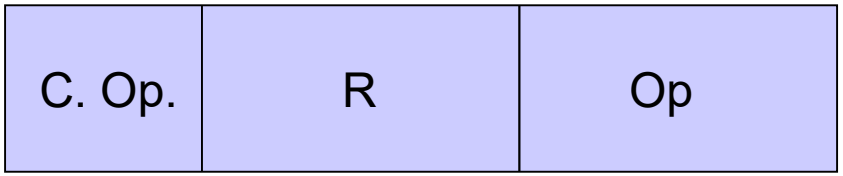
Exemplos de formatos de instruções de máquina.



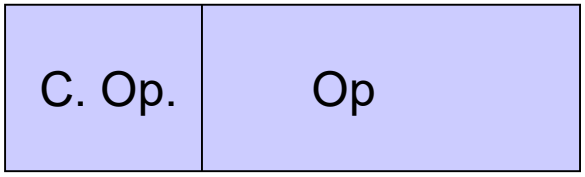
ADD Op1, Op2, Op3



MOVE Op1, Op2



ADC R, Op



JCXZ Op

Quantidade de operandos

- Um dos fatores mais importantes no projeto de uma UCP consiste na escolha do tamanho do tamanho das instruções.

- ✓ Tamanho da memória
- ✓ Tamanho e organização das células da MP;
- ✓ Velocidade de acesso;
- ✓ Organização do barramento de dados.

Instruções com 3 operandos

- No caso de operações aritméticas com dois valores, sua execução requer uma indicação explícita da localização desses valores.

ADD A,B,X (X)←(A) + (B)

SUB A,B,X (X)←(A) - (B)

MPY A,B,X (X)←(A) x (B)

DV A,B,X (X)←(A) / (B)

Instruções com 2 operandos

- Considerando a importância do problema de economia de espaço de armazenamento, foram criadas instruções com dois campos de operandos.

ADD A,B (A)←(A) + (B)

SUB A,B (A)←(A) - (B)

MPY A,B (A)←(A) x (B)

DV A,B (A)←(A) / (B)

Instruções com 1 operando

- Com esse tipo, o acumulador (ACC) é empregado como operando implícito, ou seja não é necessário especificar seu endereço na instrução.

ADD Op. ACC ← ACC + Op.

SUB Op. ACC ← ACC - Op.

MPY Op. ACC ← ACC x Op.

DIV Op. ACC ← ACC / Op.

Modo de endereçamento

- É a forma de sinalizar a localização de um dado.
- A existência de vários métodos para localizar um dado que está sendo referenciado em uma instrução se prende à necessidade de dotar os sistemas de computação da necessária flexibilidade no modo de atender aos diferentes requisitos dos programas.

Modos de endereçamento

- Imediato;
- Direto;
- Indireto;
- Por registrador;
- Indexado;
- Base mais deslocamento.

CPU

- Exemplos:
  - ADD A,B,X (X recebe A+B)
  - MUL R1,R2,R3 (R3 recebe R1*R2)
  - ADD A,B (A recebe A+B)
  - SUB R4,R2 (R4 recebe R4 – R2)
  - ADD A (Acc recebe Acc+A)
  - DIV R5 (Acc recebe Acc+R5)

CPU

- Modo de endereçamento:
  - Nem sempre e valor do operando é o que deve ser utilizado para a realização da operação;
  - Às vezes, o valor do operando significa onde o dado real se encontra na memória principal do computador;
  - É preciso definir qual o modo de endereçamento dos dados;

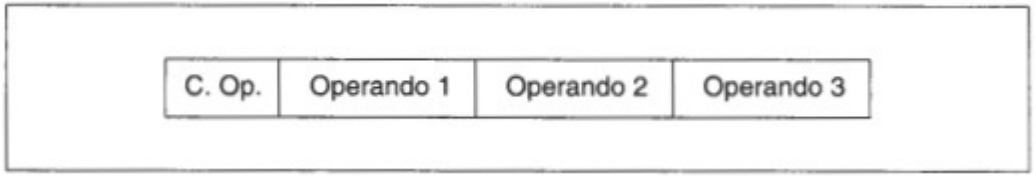
CPU

- Endereçamento imediato:
  - O valor do operando é exatamente o que deve ser utilizado pela CPU;
  - Ex: ADD 3
  - (Acc recebe Acc+3)
  - Vantagem: como o dado já está disponível, não é necessário buscá-lo na memória, diminuindo o tempo de processamento;
  - Desvantagem: a quantidade de bits geralmente será pequena, pois é necessário reservar alguns bits para o código de operação

CPU

- Endereçamento indireto:
  - O valor do operando indica onde está na memória o endereço do dado real;
  - Ex: DIV A
  - (Acc recebe o valor da divisão de Acc pelo valor contido no endereço A da memória);
  - Vantagem: não há limitação tão restrita do espaço de memória;
  - Desvantagem: é preciso acessar a memória duas vezes;

INSTRUÇÕES COM 3 OPERANDOS



ADD A,B,X	(X) ← (A) + (B)
SUB A,B,X	(X) ← (A) − (B)
MPY A,B,X	(X) ← (A) (B)
DIV A,B,X	(X) ← (A) (B)

EXEMPLO DE OPERADORES

Para exemplificar sua utilização, consideremos que um programa escrito em linguagem de alto nível contenha o comando mostrado a seguir, o qual calcula o valor de uma expressão algébrica:

$X = A * (B + C * D - E/F)$

Como resultado do processo de compilação (ver Cap. 9), o referido comando será convertido em instruções de máquina que, em conjunto, representam um programa com resultado idêntico ao do comando já apresentado.

Para simplificar e melhorar nosso entendimento, vamos utilizar as instruções da linguagem Assembly para montar o programa equivalente, em vez de criarmos instruções em linguagem binária:

A seqüência do algoritmo para resolver a equação é a seguinte, considerando as regras matemáticas usuais:

1) Inicialmente, resolver as operações internas aos parênteses.
2) Dentre as operações existentes, a primeira a ser realizada é a multiplicação de C por D (o resultado é armazenado em uma variável temporária, T1) e, em seguida, a divisão de E por F (resultado em uma variável temporária, T2) — prioridade dessas operações sobre soma e subtração.
3) Posteriormente, efetua-se a soma de B com T1.
4) Subtrai-se T2 do resultado dessa soma.
5) Finalmente, multiplica-se A por esse último resultado e armazena-se em X.

EXEMPLO COM TRÊS OPERANDOS

$$X = A * (B + C * D - E/F)$$

```
MPY     C,D,TI      ; multiplicação de C e D, resultado em T1 (item 2)
DIV     E,F,T2      ; divisão de E por F, resultado em T2 (item 2)
ADD     B,TI,X      ; soma de B com T1; resultado em X (item 3)
SUB     X,T2,X      ; subtração entre X e T2, resultado em X (item 4)
MPY     A,X,X       ; multiplicação de A por X, resultado em X (item 5)
```

INSTRUÇÕES COM DOIS OPERANDOS

ADD A,B (A) ← (A) + (B)

ADD	Op.1,Op.2	(Op.1) ← (Op.1) + (Op.2)
SUB	Op.1,Op.2	(Op.1) ← (Op.1) − (Op.2)
MPY	Op.1,Op.2	(Op.1) ← (Op.1) * (Op.2)
DIV	Op.1,Op.2	(Op.1) ← (Op.1) /(Op.2)

EXEMPLO DE OPERADORES

Para exemplificar sua utilização, consideremos que um programa escrito em linguagem de alto nível contenha o comando mostrado a seguir, o qual calcula o valor de uma expressão algébrica:

$X = A * (B + C * D - E/F)$

Como resultado do processo de compilação (ver Cap. 9), o referido comando será convertido em instruções de máquina que, em conjunto, representam um programa com resultado idêntico ao do comando já apresentado.

Para simplificar e melhorar nosso entendimento, vamos utilizar as instruções da linguagem Assembly para montar o programa equivalente, em vez de criarmos instruções em linguagem binária:

A seqüência do algoritmo para resolver a equação é a seguinte, considerando as regras matemáticas usuais:

1) Inicialmente, resolver as operações internas aos parênteses.
2) Dentre as operações existentes, a primeira a ser realizada é a multiplicação de C por D (o resultado é armazenado em uma variável temporária, T1) e, em seguida, a divisão de E por F (resultado em uma variável temporária, T2) — prioridade dessas operações sobre soma e subtração.
3) Posteriormente, efetua-se a soma de B com T1.
4) Subtrai-se T2 do resultado dessa soma.
5) Finalmente, multiplica-se A por esse último resultado e armazena-se em X.

EXEMPLO COM DOIS OPERANDOS

$$X = A * (B + C * D - E/F)$$

```
MPY     C,D,TI      ; multiplicação de C e D, resultado em T1 (item 2)
DIV     E,F,T2      ; divisão de E por F, resultado em T2 (item 2)
ADD     B,TI,X      ; soma de B com T1; resultado em X (item 3)
SUB     X,T2,X      ; subtração entre X e T2, resultado em X (item 4)
MPY     A,X,X       ; multiplicação de A por X, resultado em X (item 5)
```

EXEMPLO COM DOIS OPERANDOS

$$X = A * (B + C * D - E/F)$$

```
MPY     C,D     ; multiplicação de C por D, resultado em C (item 2)
DIV     E,F     ; divisão de E por F, resultado em E (item 2)
ADD     B,C     ; soma de B com C, resultado em B (item 3)
SUB     B,E     ; subtração entre B e E, resultado em B (item 4)
MPY     A,B     ; multiplicação de A por B, resultado em A (item 5)
MOVE    X,A     ; armazenamento do resultado final, A, em X
```

EXEMPLO COM UM OPERANDO

ADD Op.	ACC ← ACC + (Op.)
SUB Op.	ACC ← ACC − (Op.)
MPY Op.	ACC ← ACC (Op.)
DIV Op.	ACC ← ACC (Op.)

TAMANHO E CONSUMO DE TEMPO DE EXECUÇÃO DE INSTRUÇÕES

Instrução de 3 operandos — C.Op. = 8 bits + 3 operandos de 20 bits cada um = 68 bits
Ciclos de memória = 4 (um para buscar a instrução e 3 para cada operando)

Instrução de 2 operandos — C.Op. = 8 bits + 2 operandos de 20 bits cada um = 48 bits
Ciclos de memória = 4 (um para buscar a instrução e 3 para cada operando)

Instrução de 1 operando — C.Op. = 8 bits + 1 operando de 20 bits = 28 bits
Ciclos de memória = 2 (um para buscar a instrução e 1 para o operando)

EXEMPLO COM TRÊS INSTRUÇÕES DE OPERANDO

$$X = A * (B + C * D - E/F)$$

```
MPY     C, D, T1
DIV     E, F, T2
ADD     B, T1, X
SUB     X, T2, X
MPY     A, X, X
```

EXEMPLO COM DOIS INSTRUÇÕES DE OPERANDO (SEM SALVAMENTO)

$$X = A * (B + C * D - E/F)$$

```
MPY     C, D
DIV     E, F
ADD     B, C
SUB     B, E
MPY     A, B
MOVE    X, A
```

EXEMPLO COM DOIS INSTRUÇÕES DE OPERANDO (COM SALVAMENTO)

$$X = A * (B + C * D - E/F)$$

```
MOVE    X, C
MPY     X, D
MOVE    T1, E
DIV     T1, F
ADD     X, B
SUB     X, T1
MPY     X, A
```

EXEMPLO COM UMA INSTRUÇÃO DE OPERANDO

$$X = A * (B + C * D - E/F)$$

```
LDA   C
MPY   D
STA   X
LDA   E
DIV   F
STA   T1
LDA   B
ADD   X
SUB   T1
MPY   A
STA   X
```

Com o propósito de permitir a transferência de dados entre o ACC e a MP, foram criadas duas novas instruções:

LDA Op.	que significa	ACC ← (Op.)
STA Op.		(Op.) ← ACC

EXEMPLOS

$$X = A * (B + C * D - E/F)$$

Com instruções de 3 operandos	**Com instruções de 2 operandos (sem salvamento)**	**Com instruções de 2 operandos (com salvamento)**	**Com instruções de 1 operando**

MPY C, D, T1
DIV E, F, T2
ADD B, T1, X
SUB X, T2, X
MPY A, X, X
	MPY C, D
	DIV E, F
	ADD B, C
	SUB B, E
	MPY A, B
	MOVE X, A
		MOVE X, C
		MPY X, D
		MOVE T1, E
		DIV T1, F
		ADD X, B
		SUB X, T1
		MPY X, A
			LDA C
			MPY D
			STA X
			LDA E
			DIV F
			STA T1
			LDA B
			ADD X
			SUB T1
			MPY A
			STA X

Espaço: 340 bits Tempo: 20 acessos	Espaço: 288 bits Tempo: 24 acessos	Espaço: 336 bits Tempo: 28 acessos	Espaço: 308 bits Tempo: 22 acessos

REALIZE AS SEGUINTES OPERAÇÕES COM 3, 2 e 1 INSTRUÇÃO

a) X = A + (B * (C − A) + (D − E/B) * D)

b) Y = (A + B * (C − D * (E/(B − F)) + B) * E)

CONSIDERE AS INSTRUÇÕES DEFINIDAS A SEGUIR DE UM OPERANDO

LDA Op.	ACC ← (Op.)	STA Op.	(Op.) ← ACC
ADD Op.	ACC ← ACC + (Op.)	SUB Op.	ACC ← ACC − (Op.)
MPY Op.	ACC ← ACC * (Op.)	DIV Op.	ACC ← ACC/(Op.)

Deduza a equação matemática cuja solução resultou no seguinte programa, criado com estas instruções:

```
LDA   A
ADD   C
STA   X
LDA   B
MPY   D
SUB   E
STA   Y
LDA   X
ADD   Y
DIV   F
STA   X
```