Organização de Computadores

Aula 07: Processador e Conjunto de Instruções

INTRODUÇÃO

Agora que já conhecemos o funcionamento de um processador de acordo com o modelo de Von Neumann e o ciclo de instrução, estudaremos a execução dessas instruções.

Nesta aula, será relembrado que, ao se desenvolver um programa em linguagem de alto nível, ele precisa ser traduzido para linguagem de máquina, de forma que instruções sejam compatíveis com aquele processador, uma vez que todos eles são fabricados para conter um conjunto de instruções compatíveis com o mesmo.

Veremos que, para cada tipo de instrução, existem dois agentes principais, operador e operando, responsáveis pelo direcionamento da instrução a ser executada, de acordo com seu grau de complexidade.

Posteriormente, estudaremos que existem formas de otimização na execução de instruções, através do Paralelismo, que é considerado um método aplicado hoje nos processadores para otimizar o uso de qualquer possibilidade de ociosidade em um ciclo de instrução.

OBJETIVOS

Reconhecer as instruções de máquina e suas funcionalidades.

Distinguir os tipos de instrução de acordo com a quantidade de operandos usados em sua formação e o que isso influencia na execução.

Interpretar paralelismo a nível de instrução e seu uso como método de otimização do desempenho da CPU.

INTRODUÇÃO

Uma vez conhecido o funcionamento de um processador de acordo com o modelo de Von Neumann e do ciclo de instrução, é importante entender as instruções.

Todo programa desenvolvido precisa ser traduzido para linguagem de máquina, de forma que as instruções sejam compatíveis com aquele processador, uma vez que todo o processador contém um conjunto de instruções relacionado à capacidade de executar.

INSTRUÇÃO DE MÁQUINAS

Como já estudado, quem executa um programa e suas instruções é o hardware, que, através dos ciclos de instrução do processador, busca as instruções em linguagem de máquina.

Afinal, um programa desenvolvido em linguagem de Alto Nível (mais amigável ao programador) não pode ser executado diretamente pelo hardware. Ele precisa ser traduzido para linguagem de máquina por um compilador antes de ser efetivamente carregado em memória, a fim de que o processador possa executá-lo.

A linguagem de máquina é composta de códigos binários, representando essas instruções, endereços e dados e está totalmente vinculada ao conjunto de instruções definido para uma máquina.

Em termos de funcionalidade, as operações da máquina podem ser:

Matemáticas	Ex: Operações aritméticas, lógicas, de complemento.
Movimentação de dados	Ex: Movimentação da memória para um registrador.
Entrada/saída	Ex: Leitura e gravação de dados em dispositivos externos.
Controle	Ex: Desvio condicional da sequência de execução (Se X>18 então ...).

FORMATO DAS INSTRUÇÕES

Operando
Define o valor binário da localização do
dado ou mesmo o próprio conteúdo do dado a ser processado pela instrução
definida em seu código de operação.
Normalmente o operando define um endereço
de memória que possui o dado armazenado ou onde será armazenado o resultado de
uma operação. Operações aritméticas possuem em geral 2 ou 3 operandos.

Código de operação	**Operando 1**	**Operando 2**	**Operando 3**

Código de Operação
Define a operação que será realizada pelo
processador. É o campo da instrução onde o valor binário identifica a operação
a ser realizada
Este código é a entrada no Decodificador de
Instruções para a unidade de controle. Cada instrução possui um código único,
que será seu identificador dentro da execução de instruções em um processador.

Aspectos dos Conjuntos de Instruções

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.

O conjunto de instruções normalmente pode ser analisado sob alguns aspectos, como, por exemplo:

Quantidade de instruções → **Quantidade de operandos (normalmente 1, 2 ou 3 operandos)** → **Modo de endereçamento**

Veja o exemplo de uma instrução com 2 operandos:

ADD	SALÁRIO	TAXA
11010011	**110011001100**	**101010101010**
8 BITS	**12 BITS**	**12 BITS**

Podemos ver que na formação da instrução temos:

A Operação ADD definida por um código binário de 8 bits → **Somado a 2 Operandos, separados por vírgulas e definidos por 12 Bits cada um**

Totalizando 32 Bits de uma instrução

Isso já nos leva a entender que a Palavra desse processador seria de 32 Bits, que é o tamanho da informação que trafega entre a CPU e a memória principal. Dessa forma, temos um exemplo de uma instrução definida por 1 operação e 2 operandos.

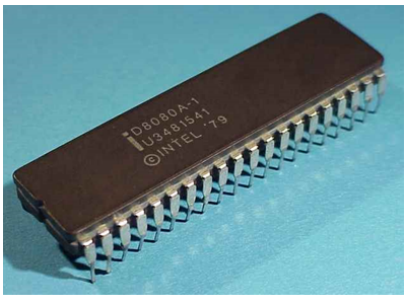
Veremos posteriormente exemplos práticos de instruções com 1, 2 e 3 Operandos, o que permite verificar como é o processamento interno de uma máquina, por exemplo, na execução de uma expressão matemática.

Curiosidade
, O conjunto de instruções definido em um processador é a base para que um programa seja executado nesse processador.

Um exemplo interessante é o processador Intel 8080, primeiro processador comercial da Intel.

Ele não possuía instruções para multiplicação ou divisão. Sendo assim, um programa em linguagem de máquina, neste processador, não conseguiria efetuar operações de multiplicação ou divisão se não fosse pela combinação de outras instruções como, por exemplo, múltiplas somas para executar uma multiplicação.

Porém, quando desenvolvido em uma linguagem de alto nível, a mesma poderia implementar comandos de multiplicação e divisão que já aplicassem uma série de instruções repetidas de soma ou subtração no momento da compilação. , ,



//www.antiquetech.com/wp-content/uploads/2013/09/D8080A-1.jpg

QUANTIDADE DE OPERADOS

Instruções de máquinas são constituídas por um conjunto de beta, que contém um subconjunto chamado código de operação, responsável por identificar a operação a ser realizada pelo hardware.

Esse código é decodificado na UC na fase de instrução, gerando os pulsos de controle para acionar as portas lógicas necessárias à execução da operação.

Possui ainda um ou mais grupos de bits denominados campo(s) do(s) operando(s) que tem por função identificar e localizar o dado a ser processado.

Instruções com 3 operandos

Dentro de uma instrução de 3 operandos, os campos 1 e 2 representam os endereços de cada dado que será utilizado na operação.

O campo relativo ao Operando 3 contém o endereço onde será armazenado o resultado da operação em execução.

Como exemplo, teríamos como operações fundamentais (ADD = SOMA, SUB = SUBTRAÇÃO, MPY = MULTIPLICAÇÃO e DIV = DIVISÃO) as representações abaixo:

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

Quando verificar as instruções acima, pense da seguinte forma:

ADD	A, B, X	➡	“Pegue A, some com B, e armazene em X”

Seguindo esse pensamento, como seria escrito um conjunto de instruções de 3 operandos para executar o cálculo da expressão abaixo?

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

Resultado:

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

1	MPY	C,D,T1
2	DIV	E,F,T2
3	ADD	B,T1,X
4	SUB	X,T2,X
5	MPY	A,X,X

Veja que com 3 operandos foram necessárias 5 linhas de programação.

Respeitando as regras matemáticas e priorizando parênteses, multiplicações e divisões, a ordem criada na programação das instruções foi a seguinte:

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

1	MPY	C,D,T1	Ficamos com X = A*(B+T1-E/F)
2	DIV	E,F,T2	Ficamos com X = A*(B+T1-T2)
3	ADD	B,T1,X	Ficamos com X = A*(X-T2)
4	SUB	X,T2,X	Ficamos com X = A*(X)
5	MPY	A,X,X	Ficamos com X = X

Feito isso, o cálculo foi realizado utilizando-se de 2 endereços de memória adicionais (T1 e T2, além de X, que era o endereço final para armazenamento do cálculo da expressão.

Instruções com 2 operandos

Nas instruções com 2 operandos, novas operações aparecem, em virtude de algumas limitações. Quando há somente 2 operandos, o campo 1 sempre recebe o resultado da operação.

ADD	A, B	➡	A=A+B

SUB	A, B	➡	A=A-B
MPY	A, B	➡	A=A*B
DIV	A, B	➡	A=A/B
MOVE	A, B	➡	A=B

Seguindo esse pensamento, como seria escrito um conjunto de instruções de 2 operandos para executar o cálculo da mesma expressão que foi feita anteriormente?

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

Resultado:

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

1	MPY	C,D
2	DIV	E,F
3	ADD	B,C
4	SUB	B,E
5	MPY	A,B
6	MOVE	X,A

Com a mudança de operandos, também foi alterada a quantidade de linhas de código necessárias para efetuar o cálculo, de forma que sua ordem de execução foi a seguinte:

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

1	MPY	C,D	Ficamos com X = A*(B+C-E/F)

2	DIV	E,F	Ficamos com X = A*(B+C-E)
3	ADD	B,C	Ficamos com X = A*(B-E)
4	SUB	B,E	Ficamos com X = A*(B)
5	MPY	A,B	Ficamos com X = A
6	MOVE	X,A	Ficamos com X = X

Perceba que a Operação Move, como o próprio nome diz, MOVE o conteúdo de um endereço de memória, para outro endereço. Sendo assim, o conteúdo que estava no endereço de memória A foi movido para o endereço X.

Instruções com 1 operando

Nas instruções com 1 operando, o uso do Acumulador (registrador utilizado para armazenamento temporário na CPU) é constante, pois é utilizado como um operando não expressamente declarado, mas utilizado em todas as operações.

ADD	Op.	➡	ACC=ACC+Op
SUB	Op.	➡	ACC=ACC-Op
MPY	Op.	➡	ACC=ACC*Op
DIV	Op.	➡	ACC=ACC/Op
LDA	Op.	➡	ACC=Op
STA	Op.	➡	Op=ACC

Nesse tipo de execução, tem-se 2 novas operações, sendo elas LDA (LOAD) e STA (STORAGE), onde:

- **LDA se refere a carregar um conteúdo armazenado na memória; e**
- **STA a armazenar um conteúdo na memória**

Quando visualizadas as operações, implicitamente o Operando ACC deve ser considerado, ou seja, no exemplo ADD Op., entende-se:

“Pegue o conteúdo do Acumulador, some ao Operando, e grave novamente no Acumulador”.

Seguindo esse pensamento, como seria escrito um conjunto de instruções de 2 operandos para executar o cálculo da mesma expressão que foi feita anteriormente?

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

Resultado:

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

1	LDA C	Carrega C no Acc
2	MPY D	Efetua Acc*D e grava em Acc
3	STA X	Armazena o conteúdo de Acc em X
4	LDA E	Carrega E no Acc
5	DIV F	Efetua Acc/F e grava em Acc
6	STA T1	Armazena o conteúdo de Acc em T1
7	LDA B	Carrega B no Acc
8	ADD X	Efetua Acc+X e grava em Acc
9	SUB T1	Efetua Acc-T1 e grava em Acc
10	MPY A	Efetua Acc*A e grava em Acc
11	STA X	Grava o conteúdo de Acc em X

Veja que, durante a execução das instruções, ele preparou C*D e gravou em X, como também E/F e gravou em T1, posteriormente, executando uma operação após a outra. Mesmo assim, foram necessárias 11 linhas de operações para a execução da mesma expressão matemática.

MODOS DE ENDEREÇAMENTO DE INSTRUÇÕES

Sabe-se que os programas a serem executados e, consequentemente, suas instruções são carregados na memória principal.

Para que o conteúdo a ser lido ou armazenado seja referenciado, é necessário ter um modo de endereçamento que possa identificá-lo individualmente para que a unidade central de processamento, ou CPU, possa determinar qual posição de memória está sendo usada por uma instrução da máquina.

Sendo assim, existem alguns modos de endereçamento:

• **Imediato:** utiliza um valor como operando e não um endereço na memória, ou seja, não é utilizado um endereço da memória, o operando é parte da instrução. Permite definir constante e valores de variáveis iniciais;
• **Direto:** indica o endereço de memória onde está o operando;
• **Indireto:** Indica um ponteiro de endereço para um operando, onde o conteúdo deste ponteiro não é um dado e sim outro endereço. Há um duplo endereçamento neste caso;
• **Por registrador:** O endereço se refere a um endereço de registrador e não da memória principal;
• **Base + Deslocamento:** Endereço obtido pela soma do operando com o conteúdo de um registrador-base.

PARALELISMO

Como temos estudado desde o início, o resultado de toda a evolução na computação é a busca pela otimização.

Todas as evoluções conhecidas até o momento são resultado dessas buscas. Um destes resultados, perceptíveis e em constante crescimento, é a frequência de operação das máquinas, ou clock interno dos processadores.

Aumentar a velocidade do clock interno dos processadores sempre é sinônimo de aumentos significativos na performance de um equipamento. Porém, existem outros métodos, desenvolvidos no decorrer das pesquisas, que permitiram a evolução na otimização do ciclo de Busca–Decodificação–Execução, conhecida como Paralelismo.

O Paralelismo, como a própria palavra diz, é a possibilidade de executar instruções em paralelo. Quando pensamos desta forma, em um primeiro momento, pensamos nos atuais processadores Dual Core, Quad Core, entre outros, que não deixam de ser um tipo de paralelismo a nível de Processador.

E se dissermos que há possibilidade de paralelismo em um único processador?

É isso que veremos a partir de agora.

Formas gerais de paralelismo

Existem duas formas gerais de paralelismo:

• Paralelismo no nível de instrução ➡ **• Paralelismo no nível de processador**

Paralelismo no nível de instrução

Dos diversos métodos de paralelismo a nível de instrução, o mais conhecido deles é o PIPELINE.

Pipeline foi uma técnica revolucionária que consistia em dividir a execução de uma instrução através do ciclo de instruções em diversas etapas, chamadas estágios, onde cada uma era manipulada por um agente distinto dentro do ciclo de instrução.

Dessa forma, era possível identificar agentes “ociosos” dentro do processo, podendo agilizar e iniciar outra instrução, mesmo que a anterior não tivesse sido finalizada.

Tomemos como base o ciclo de Busca–Decodificação–Execução e algumas de suas etapas:

Carrega Instrução [C]	➜	Acessa Memória [M]	➜	Decodifica Instrução [D]	➜	Executa Instrução [X]	➜	Armazena resultado [W]

Fonte: Adaptado pelo Autor

Se tomarmos originalmente o ciclo de instrução aprendido e o ciclo de clock, a próxima instrução seria carregada somente depois que TODO o ciclo fosse executado. Porém, entende-se que, se a instrução foi carregada e já está no processo de busca na memória, o processo poderia ser otimizado.

O Pipeline foi criado para isso. Trata-se de uma metodologia básica para melhorar a velocidade, reduzindo o número de ciclos de clock necessários para executar uma instrução, permitindo sobrepor a execução das instruções através dos pipelines, fazendo, assim, o uso uso mais adequado do hardware.

Veja o esquema abaixo, de acordo com a cronologia do tempo:

Tempo	1	2	3	4	5	6	7	8	9	10	11	12
	[C]	[M]	[D]	[X]	[W]							
		[C]	[M]	[D]	[X]	[W]						
			[C]	[M]	[D]	[X]	[W]					
				[C]	[M]	[D]	[X]	[W]				
					[C]	[M]	[D]	[X]	[W]			
						[C]	[M]	[D]	[X]	[W]		
							[C]	[M]	[D]	[X]	[W]	
								[C]	[M]	[D]	[X]	[W]

Perceba que, com o pipeline, ocorre uma independência entre as fases de execução da instrução.

Não será sempre o caso, mas, em boa parte das vezes, o pipeline evitará a ociosidade e aumentará a eficiência do processador.

Se analisar, no tempo 5, há a primeira instrução no final de execução. Contudo, outras 4 instruções já foram iniciadas, simplesmente usando a ociosidade das etapas anteriores do ciclo de instrução.

Logo, o paralelismo a nível de instrução, através do pipeline, aumenta o desempenho de uma CPU devido ao aumento da vazão das instruções, para que haja maior número de instruções executadas por unidade de tempo.

Atenção
, Importante lembrar que o tempo de execução de cada instrução permanece o mesmo. Porém, as mesmas são antecipadas, permitindo que o tempo final de execução de um conjunto de instruções seja menor quando usado o pipeline.

ATIVIDADE

Imagine que você é uma CPU, porém não conhece instruções para multiplicação e divisão de números. Como poderia calcular uma multiplicação ou divisão, sem possuir essas operações?

Resposta Correta

EXERCÍCIOS

Questão 1: Considerando uma instrução com código de operação de 4 bits e operando de 8 bits, quantas instruções no máximo podem existir nessa configuração?

- ○ 16 instruções diferentes
- ○ 8 instruções diferentes
- ○ 64 instruções diferentes
- ○ 4 instruções diferentes
- ○ 32 instruções diferentes

Justificativa

Questão 2: Considere a expressão X = A*(B+C*D-E/F). O conjunto de instruções de 3 operandos abaixo representa as instruções necessárias para a aplicação da mesma.

01: MPY C,D,T1
02: DIV E,F,T2
03: ADD B,T1,X
04: _____
05: MPY A, X,X

Identifique a instrução na linha 04 para que o conjunto de instruções obtenha o resultado correto da expressão:

- ○ ADD X,T2,X
- ○ SUB X,T2,X
- ○ MPY T1,T2,X
- ○ SUB X,T2,T1
- ○ SUB X,X,T2

Justificativa

Questão 3: São Modos de Endereçamento de Instruções, EXCETO:

- ○ Imediato
- ○ Direto
- ○ Indireto
- ○ Por registrador
- ○ Condicional

Justificativa

Glossário