Unidade: Tópicos de Arquitetura de computação

Unidade: Tópicos de Arquitetura de computação

1) INSTRUÇÕES

Cada processador tem um conjunto de instruções que define como ele irá manipular os dados. Para programar em linguagem de máquina, deve-se conhecer como as instruções são representadas e, a partir daí, traduzir um algoritmo para linguagens de baixo nível, como o *assembly*. Nesta aula, você irá estudar sobre as características de uma instrução e o conjunto de instrução ISA.

1.1) Instruções de máquina

A forma como a CPU funciona é determinada pelas instruções que ela executa. Como já descrito antes, os microprocessadores tem um conjunto de instruções pré-determinadas, conhecidas como instruções de máquina ou *siaa* do computador. O conjunto de diferentes instruções que a CPU é capaz de executar é conhecida como conjunto de instruções da CPU.

1.1.1) **Elementos de instruções de máquinas**

Cada instrução é composta por um conjunto de *bits* e deve conter todas as informações necessárias para que a CPU possa executá-las através de seus circuitos digitais combinacionais internos.

Basicamente, os elementos de instruções de máquinas são:

- **Código de operação:** especifica a operação a ser efetuada (por exemplo, ADD, SUB etc). A operação é especificada por um código binário, conhecido como código de operação;
- **Referência ao operando fonte:** a operação pode envolver um ou mais operando, ou seja, os operandos que constituem dados de entrada para a operação;
- **Referência ao operando de destino:** a operação pode produzir um resultado;
- **Endereço da próxima instrução:** indica onde a CPU deve buscar a próxima instrução, depois que a execução da instrução corrente for completada.

A próxima instrução a ser buscada e executada pode estar localizada na

memória principal ou, no caso de um sistema com memória virtual, tanto na memória principal quanto na memória secundária (HD). Na grande maioria dos casos, a próxima instrução na memória principal está imediatamente após a instrução corrente. Nesses casos, até para melhorar o desempenho, a instrução não inclui uma referência explícita para a próxima posição de memória. Quando uma instrução não está subsequente à instrução corrente, a instrução deve fornecer o endereço de memória principal ou de memória virtual.

1.2) Representação de instruções

Internamente, cada instrução de um computador é representada como uma sequência de *bits*. Uma instrução é dividida em campos, correspondentes aos elementos da instrução. Para a maioria dos conjuntos de instruções é usado mais de um formato de instrução. Durante a execução, conforme já descrito em aulas anteriores, uma instrução é lida pelo registrador de instruções (IR) da CPU. A CPU deve ser capaz de extrair os dados dos vários campos da instrução e efetuar a operação requerida.

4 bits	6 bits	6 bits
Código de operação	Referência a operando	Referência a operando

Figura1: Formato de instrução simples

Perceba que, somando a quantidade de *bits* do formato de uma instrução simples, chegamos ao valor de 16 *bits*. É difícil para um programador lidar com representações binárias de instruções de máquinas para desenvolver um programa. Por isso, tornou-se prática comum usar uma representação simbólica para instruções de máquina, isto significa que para cada instrução que o microprocessador pode executar, há um “apelido” dado a ela para facilitar a sua interpretação. Portanto, os códigos de operação são representados por abreviações, chamados *mnemônicos*, que indicam a operação a ser efetuada. Alguns exemplos comuns são:

- ADD-adição;
- SUB-subtração;
- MPY-multiplicação;
- DIV-divisão;
- LOAD-carregar dados da memória;
- STOR-armazenar dados na memória.

Os operandos são também representados de maneira simbólica. Por exemplo, a instrução.

ADD R, Y

Pode significar adicionar o valor contido na posição Y no registrador R. Neste exemplo, Y é o endereço de uma posição de memória e R indica um registrador. Lembre-se de que os registradores estão dentro do microprocessador e são pequenas posições de memória que têm a função de armazenar dados para serem processados, ou então enviados para memória principal. No exemplo acima, a operação é feita sobre o conteúdo da posição de memória, e não sobre seu endereço; isto significa que estamos indicando um endereço onde será considerado o conteúdo deste endereço para ser somado com o conteúdo que está armazenado no registrador R.

Portanto, como você já deve ter notado, é possível escrever um programa em linguagem de máquina usando símbolos que representam as instruções em binário. Cada código de operação simbólico tem uma representação binária correspondente, e o programador especifica o endereço de cada operando simbólico. Por exemplo, o programador pode começar com a seguinte lista de definições:

X=513
Y=514

Como cada símbolo tem a sua representação binária, depois de escrito todo o programa na forma simbólica, ele deve ser convertido em binário. A conversão pode ser feita por meio de um programa bastante simples, resultando em instruções de máquina binárias.

Hoje, é muito raro programar diretamente em linguagem de máquina. A maioria dos programas é escrito em linguagem de alto nível ou, em alguns casos bem específicos, linguagens de montagem. Muitos ainda recorrem à linguagem C e usam conversores para obter o equivalente em linguagem de máquina.

1.3) Conjunto de Instruções – nível ISA (*Instruction Set Architecture*)

A arquitetura do conjunto de instruções é a interface entre a linguagem de programação e o *hardware* que executam os programas. O ISA (*Instruction Set Architecture*) especifica o tamanho da memória principal, a quantidade de registradores e o número de *bits* por instrução. Especifica, também, quais as instruções de máquina que o processador é capaz de executar e como cada um dos *bits* das instruções é interpretado. Cada instrução manipula o conteúdo da memória principal, o conteúdo dos registradores ou do contador de programas. As instruções são organizadas em três categorias: aritmética, lógica, transferência entre memória e os registradores e controle de fluxo.

O ISA está posicionado entre o nível da microarquitetura e o nível do sistema operacional; esse nível foi desenvolvido antes de qualquer outro nível e, na verdade, originalmente, era o único nível disponível. O nível ISA tem um significado especial que o torna importante para arquitetos de sistema: é a interface entre o *Software* e o *Hardware*. Sendo assim, cada microprocessador desenvolvido tem que ter uma arquitetura do conjunto de instruções bem definido. Uma boa ISA deve fornecer condições claras para o código compilado.

Conforme a figura abaixo, cada instrução consiste de quatro dígitos Hexadecimais (16 *bits*). Os quatro *bits* mais a esquerda codificam um dos 16 *opcodes*, os próximos quatro *bits* referem-se a um dos 16 registradores, descritos como registradores de destino e denotados por *dest d*. A interpretação dos restantes dos *bits* mais a direita depende do *opcode*. Perceba na figura, onde está identificado como formato 1, o terceiro e o quarto dígito hexadecimal (source s e source t) são interpretados como o índice de um registrador. Por exemplo, uma instrução 1462 pode significar, adiciona o conteúdo de source s= 6 e source t= 2 e coloca o resultado em um registrador d= 4.

No *opcode* do formato 2, o terceiro e o quarto dígito hexadecimal, ou seja, os 8 *bits* mais a direita indicado como addr, são interpretados como um endereço de memória. Por exemplo, uma instrução 9462, pode significar que o conteúdo do registrador de memória d=4 tem o endereço 62.

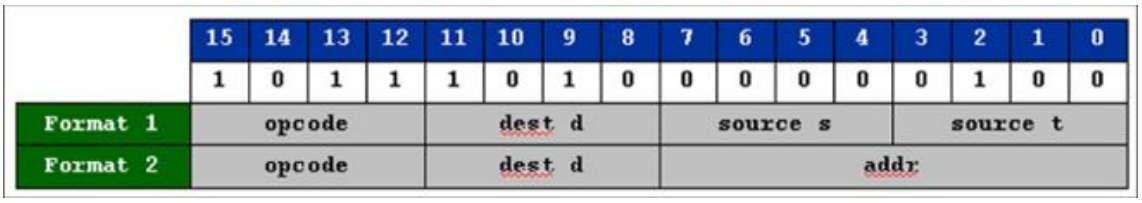

Figura 2: Formato de uma instrução.

As instruções têm um formato que deve ser entendido para que um programador possa desenvolver programas em linguagem de máquina. Dependendo da instrução, é necessário mais que 16 *bits* para representá-la; de qualquer forma, independentemente da instrução, o programador deve conhecê-la para que possa usá-la em seu código.

Os dados são armazenados em memória para uso das instruções definidas nos processadores e têm características diferentes; e estas características definem como eles serão armazenados e executados pelo conjunto de instruções disponíveis. Para executar estes diferentes tipos de dados, o processador tem diferentes formatos de instruções.

1.4) Tipos de dados

As instruções de máquina operam sobre dados, isto significa que as instruções precisam de dados para que possam ser executadas. Existem vários tipos de dados armazenados nos registradores e, na memória, cada qual ocupa um espaço diferente na memória e são tratados de acordo com o seu tipo. As classes de dados mais importantes são:

- Endereços;
- Números;
- Caracteres;
- Dados lógicos.

Algumas máquinas definem, além desses, alguns tipos de dados especiais ou estrutura de dados. Por exemplo, podem fornecer operadores que operam diretamente sobre listas ou sequência de caracteres. Vamos conhecer com mais detalhes cada um destes tipos de dados.

1.4.1) **Números**

Toda linguagem de máquina inclui tipos de dados numéricos. Mesmo o processamento de dados que não são numéricos requer o uso de números, como valor de contadores, tamanho de campos, etc. Uma distinção importante entre números usados na matemática usual e números armazenados em um computador é que esses últimos são limitados em dois sentidos. Primeiro, existe um limite para a magnitude de números representáveis em uma máquina; segundo, o caso de números de ponto flutuante, há um limite para sua precisão.

Três tipos de dados numéricos são comuns em computadores:

- Número inteiro ou de ponto fixo;
- Número de ponto flutuante;
- Número decimal.

Embora toda operação interna de um computador seja por natureza binária, os usuários do sistema lidam com números decimais. Dessa forma, números decimais devem ser convertidos para números binário na entrada, e números binário devem ser convertidos para números decimais na saída. Para aplicações com grande quantidade de entrada e saída, e relativamente pouca computação, é preferível armazenar e operar sobre números da forma decimal.

1.4.2) **Caracteres**

Uma forma comum de dado é o texto ou a sequência de caracteres. Embora dados textuais sejam convenientes para o ser humano, eles não podem ser armazenados ou transmitidos facilmente, na forma de caracteres, por sistemas de processamento de dados ou de comunicação; uma vez que os sistemas são projetados para manipular dados primários. Portanto, foram criados diversos códigos onde caracteres são representados por sequência de *bits*. O exemplo mais antigo desse tipo de código talvez seja o código Morse.

Hoje em dia, o código de caracteres mais usado é o alfabeto de referência internacional conhecido como código ASCII. Todos os caracteres desse código são representados por um padrão distinto de 7 *bits*; com 7 *bits* conseguimos representar 128 símbolos (2^7 =128). Esse número é maior que o necessário para representar os caracteres visíveis; desta forma, aproveita-se alguns padrões de *bits* para representar caracteres de controle. Alguns desses caracteres de controles são usados para controlar a impressão de caracteres em uma página, outros são usados para controlar procedimentos de comunicação. Os Caracteres ASCII são quase sempre armazenados e transmitidos usando 8 *bits* por caracteres. O oitavo *bit* pode ser sempre 0 ou pode ser usado como *bit* de paridade para detecção de erro. A detecção de erro usando paridade funciona da seguinte forma: quando configurada paridade par significa que o conjunto de 7 *bits* deve ter quantidade par de *bits* com o valor 1. Assim, se esta regra for satisfeita, então o oitavo *bit* será 0; se a regra não for satisfeita, então o valor do oitavo *bit* será 1 para tornar par a quantidade de 1 no conjunto de 8 *bits*. Se estiver configurado para paridade ímpar, então a quantidade de *bits* 1 no conjunto de 7 *bits* deve ser ímpar; se a regra for satisfeita, então o oitavo *bit* será zero; caso a regra não seja satisfeita, então o oitavo *bit* será 1.

1.4.3) **Dados lógicos**

Normalmente, cada palavra ou unidade endereçável é tratada como uma unidade única de dado. Entretanto, algumas vezes é útil considerar uma unidade de n *bits* como composta de n itens de dados, de um *bit* cada, com valor 0 ou 1. Quando o dado é visto dessa maneira, ele é considerado um dado lógico. Essa visão orientada a *bits* apresenta duas vantagens:

A primeira é a economia de memória obtida quando queremos armazenar vetores booleanos, onde cada dado apenas pode ter valor 1 (verdadeiro) ou 0 (falso). A segunda é possibilitar a manutenção de *bits* 1 de um item de dado, o que pode ser requerido em determinadas situações. Por exemplo, a implementação de operações

de ponto flutuante por *software* requer a capacidade para deslocar *bits* significativos de um operando em determinadas operações. Outro exemplo é a conversão de código ASCII para representação decimal, em que precisamos extrair os quatro *bits* mais a direita do *byte*.

Note, pelos exemplos anteriores, que o mesmo dado pode ser tratado algumas vezes como um dado lógico e, outras vezes, como um dado textual. O tipo de um dado é determinado pela operação efetuada sobre ele. Isso normalmente não ocorre na linguagem de alto nível (por exemplo, em pascal), mas quase sempre acontece em linguagem de máquina.

1.5) Formato das instruções

A memória para o microprocessador Intel 8080 é organizada em células de 8 *bits* (1 *byte*). Cada célula é identificada por um código composto por 16 *bits*, o endereço da célula, que geralmente é representado na base hexadecimal, apresenta um valor entre 0000 e FFFF. O microprocessador 8080 pode, portanto, endereçar 65.536 *bytes* de memória, compostas de memória ROM e memória de leitura / escrita.

Os dados são armazenados no microprocessador 8080 na forma de palavras com 8 *bits*, como mostra a ilustração abaixo.

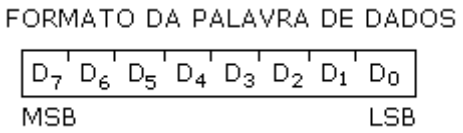

Figura 3: formato da palavra de dados

O *bit* mais à direita ou *bit* 0 é o *bit* menos significativo (em inglês, *least significant bit* ou LSB) e o *bit* mais à esquerda ou *bit* 8 é o *bit* mais significativo (em inglês, *most significante bit* ou MSB).

As instruções no microprocessador Intel 8080 podem ter 1, 2 ou 3 *bytes*; ocupando respectivamente uma, duas ou três células de memória. As ilustrações a seguir apresentam as instruções de 1, 2 e 3 *bytes*.

As instruções de 1 *byte* não tem operando.

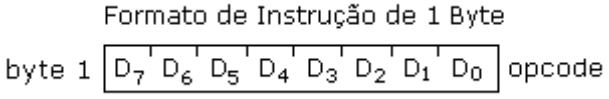

Figura 4: Formato de instrução de 1 *byte*

As instruções de 2 *bytes* tem como operando um dado de 8 *bits*.

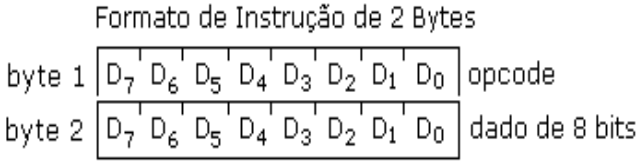

Figura 5: Formato de instrução de 2 *bytes*

As instruções de 3 *bytes* tem como operando um endereço ou um dado de 16 *bits.*

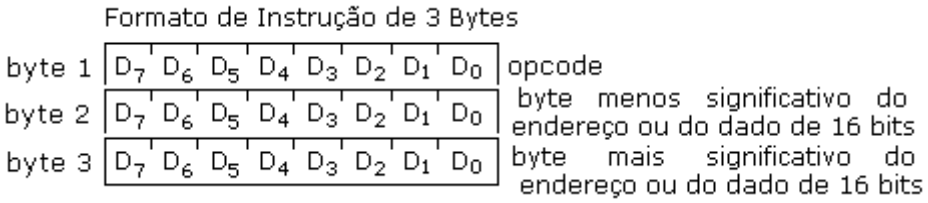

Figura 6: Formato de instrução de 3 *bytes*

Em uma instrução de 3 *bytes*, em que o operando é um endereço, o endereço é armazenado sempre com o *byte* menos significativo na posição seguinte à do *opcode*; e o *byte* mais significativo do endereço no 2.º *byte* após o *opcode*. São ocupadas 3 células consecutivas da Memória Principal do microprocessador Intel 8080, como mostra a ilustração seguinte:

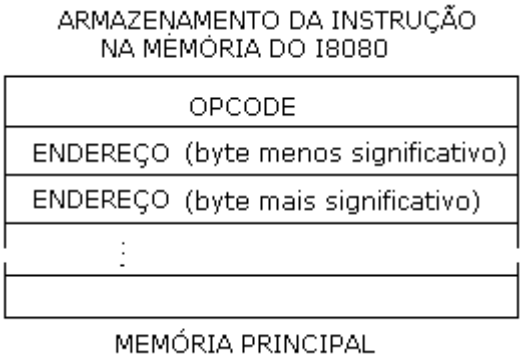

Figura 7: Armazenamento no microprocessador 8080

Em alguns casos da programação em linguagem de baixo nível (linguagem de máquina), é importante conhecer a quantidade de *bytes* que uma instrução ocupa; mesmo porque, muitas vezes, o controle da memória fica sob responsabilidade do programador; e, para este caso, deve-se calcular de forma correta para não sobrepor as informações contidas em um programa.

1.6) Ciclo de instruções

Para que você possa entender como é feita a troca de informações entre CPU e memória incluindo os registradores, vamos fazer este processo usando um processador hipotético. Na figura abaixo, são apresentados os principais componentes do processador. Como podemos ver, os registradores recebem informações e as enviam para a unidade lógica e Aritmética ou para o registrador de instrução. Mais precisamente, os dados irão para a unidade lógica e aritmética, e as instruções irão para o registrador de instruções para que elas sejam decodificadas.

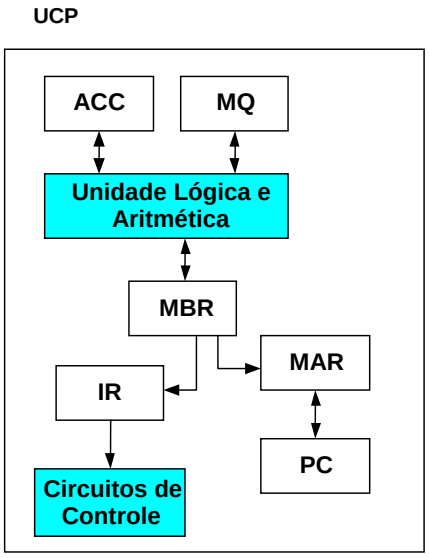

Figura 8: Componentes do processador.

Vamos agora descrever e entender a função de cada um dos componentes de um processador.

1.6.1) Registradores de Uso Geral

- **ACC (Acumulador)**: Armazena dados de origem e destino de operações da ULA. As operações realizadas são comparações entre os dados que constituem as operações lógicas e as operações de adição, subtração, divisão e multiplicação que constituem as operações aritméticas.
- **MQ (*Multiplier Quotient*)**: Utilizado em operações de multiplicação, armazenando um dos seus operandos ou os 8 *bits* menos significativos do resultado de uma multiplicação.

1.6.2) Registradores de Controle

- **MBR**: Registrador para armazenamento temporário de dados ou instruções. Neste registrador, são armazenados os dados ou instruções vindos da memória para ser enviada à unidade lógica e aritmética, ou para o

registrador de instruções. O resultado de uma operação lógica e aritmética é inserido neste registrador para, posteriormente, ser enviado para a memória. Resumindo, este registrador é o elo entre processador e a memória principal.

- **MAR**: Registrador para endereçamento de dispositivos externos (Memória, E/S, etc). Como pode perceber, este registrador é ligado ao registrador de dados, grande parte das instruções tem vinculadas a ela um endereço; quando isto ocorre, o endereço é enviado para o registrador MAR.
- **PC**: Registrador denominado Contador de Programa, este registrador é incrementado a cada ciclo de relógio. Ele pode ser alterado de acordo com as instruções vindas da memória. Este registrador reflete o endereço que deverá ser apontado na memória.
- **IR**: Registrador de Instruções, usado para receber as instruções e decodificá-las. Dentre as instruções recebidas, estão leitura da memória ou dispositivo de entrada e saída, escrita na memória ou dispositivo de saída.

1.7) DESCRIÇÃO DOS CICLOS DO COMPUTADOR

O ciclo de instrução de um processador é composto de Ciclo de Busca e de Ciclo de Instrução. A UCP tem a característica de executar uma única instrução em cada ciclo de instrução. Como todas as instruções possuem, no máximo, um operando, não existe a necessidade de haver ciclo Indireto.

A figura abaixo ilustra o ciclo de instrução da UCP e todas as partes que o constituem:

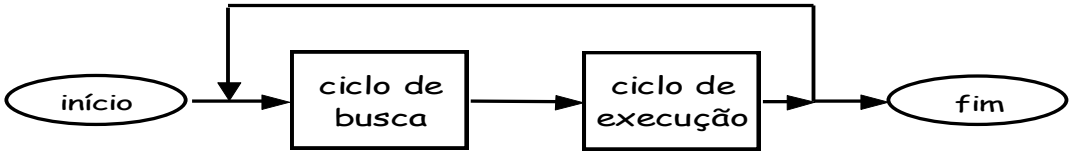

Figura 9: Ciclo de Instrução da UCP

CICLO DE BUSCA

O ciclo de Busca executa o seguinte algoritmo:

MAR ← PC;

O contador de programa é carregado para o registrador MAR. Quando ligamos o computador, o MAR é carregado com o primeiro endereço gerado pelo contador de programa (PC); na maioria das vezes, quando o computador é ligado, este endereço é o zero.

MBR ← M(MAR);

O conteúdo da memória (M) na posição indicada por (MAR) será carregado para o registrador MBR.

IR ← MBR[0:3];

O registrador de instruções (IR) recebe os 4 *bits* mais significativos de MBR, os quatro *bits* mais significativos refere-se à instrução ou *opcode*.

MAR ← MBR[4:15];

O registrador MAR recebe os *bits* da posição quatro até o décimo quinto *bit*. Estes 12 *bits* deste processador hipotético que estamos analisando, referem-se ao endereço de memória para o qual o registrador MAR irá apontar.

PC ← PC + 1;

O contador de programa é incrementado a cada pulso de relógio, ele é influenciado pelos endereços que vêm nas instruções.

Este processador, ao final da Busca, identifica qual deverá ser a instrução a ser executada e, em seguida, executa-a.

Referências

STALLINGS, W. **Arquitetura e Organização de Computadores:** Projeto Para o Desempenho. 5. ed. São Paulo: Prentice Hall, 2004.

REED, D. **Dentro do Computador** – A Arquitetura de Von Neumann. Creighton University**,** Prentice Hall. Disponível em: *http://professores.faccat.br/assis/davereed/14-DentroDoComputador.html*. Acesso em: 17 set 2010.

Edison Filho - *http://www.edisonfilho.com/Arquivos/ApostilaArqComp/A12_AC_CPU2.pdf*. Acesso em: 17 set 2010.

www.cruzeirodosul.edu.br

Campus Liberdade

Rua Galvão Bueno, 868

01506-000

São Paulo SP Brasil

Tel: (55 11) 3385-3000