Organização de Computadores

Apostila 5

Parte 2

2011

CAPÍTULO

5 UNIDADE CENTRAL DE PROCESSAMENTO

Assuntos

UD V – Unidade Central de Processamento – Parte 2

5.1 Função Controle

Objetivos

- Conhecer quais componentes da CPU exercem a Função Controle
- Conhecer como é exercida a Função Controle
- Conhecer a estrutura de uma Instrução de Máquina
- Ciclo de Instrução
- Definir o que é Pipelining
- Conhecer o que são barramentos síncrono e assíncrono
- conhecer o que é Overclocking

5.1 Função Controle

Exercida pelos componentes da CPU que se encarregam das atividades de **busca**, **interpretação** e **controle** da execução das instruções, bem como do controle da ação dos demais componentes do sistema computacional (Memória e Dispositivos E/S).

Já verificamos que as instruções de máquina que compõem um programa em execução devem estar armazenadas seqüencialmente na memória principal (e na cache, se houver uma). Já verificamos também que a UAL é o dispositivo da UCP responsável por realizar a operação matemática definida pela instrução que estiver sendo executada no momento. Falta conhecermos mais detalhes sobre as instruções de máquina, a saber:

a) o que é e como funciona uma instrução de máquina;

b) como a referida instrução de máquina é movimentada da memória para a UCP;

c) onde a instrução de máquina será armazenada na UCP; e

d) como será identificada e controlada a ação (a operação) que deve ser realizada.

A resposta para estas indagações está no conhecimento das funções desempenhadas pela área de controle de uma UCP.

A resposta para estas indagações está no conhecimento das funções desempenhadas pela área de controle de uma UCP.

A área de controle de uma UCP é a parte funcional que realiza (uma etapa de cada vez em sistemas de execução seqüencial, ou várias etapas simultaneamente, em sistemas de execução *pipelining*) as atividades de:

a) busca da instrução que será executada, armazenando-a em um registrador especialmente projetado para esta finalidade;

b) interpretação das ações a serem desencadeadas com a execução da instrução (se é uma soma, uma subtração, uma complementação etc. e como realizá-las). Em inglês, estas duas etapas compõem o que se denomina *fetch cycle*, ou ciclo de busca da instrução; e

c) geração dos sinais de controle apropriados para ativação das atividades requeridas para a execução propriamente dita da instrução identificada. Esses sinais de controle são enviados aos diversos componentes do sistema, sejam internos da UCP (como a UAL) ou externos (como a memória ou E/S). Esta etapa é denominada em inglês *execute cycle* ou ciclo de execução da instrução.

Em outras palavras, a área de controle é projetada para entender o que fazer, como fazer e comandar quem vai fazer no momento adequado. Podemos fazer uma analogia com os seres humanos, imaginando que a área de controle é o cérebro que comanda o ato de andar, e a área de processamento são os músculos e ossos das pessoas que realizam efetivamente o ato. Os nervos são análogos ao barramento de interligação entre os diversos elementos.

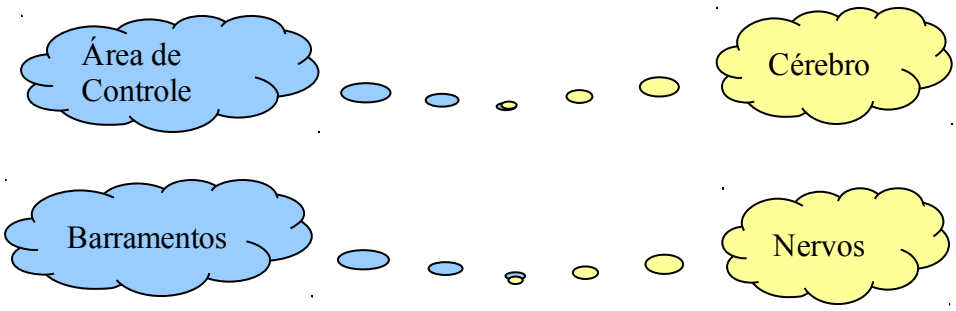
A área de controle da CPU realiza:

Fetch cycle (ciclo de busca da instrução) => busca e interpretação da instrução.

Execute cycle (ciclo de execução) => geração de sinais de controle aos diversos componentes para execução da instrução.

A área de controle é projetada para entender o que fazer, como fazer e comandar quem vai fazer, no momento adequado (quando).

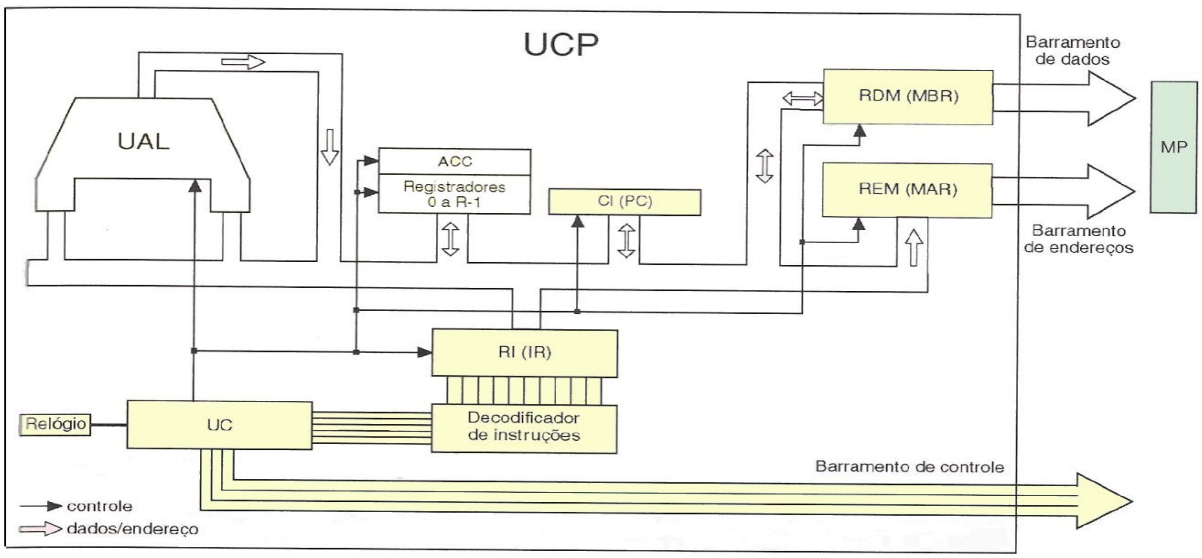
Analogia com o ser humano



Dispositivos do Processador responsáveis pela área de Controle

Os componentes da CPU responsáveis internamente pela função controle do processador estão destacados na figura e basicamente constituem-se de:

- ✔ UC – Unidade de Controle
- ✔ DI – Decodificador de Instruções
- ✔ RI – Registrador de Instrução
- ✔ CI ou PC – Contador de Instrução (*Program Counter*)
- ✔ Relógio ou *Clock*
- ✔ REM – Registrador de Endereço da Memória
- ✔ RDM – Registrador de Dados da Memória



Esquema simplificado de uma UCP

Unidade de Controle

É o dispositivo mais complexo da CPU. Possui a lógica para movimentação de dados e instruções de/para a CPU por meio de sinais de controle. Tais sinais são emitidos em vários instantes durante o período de um ciclo de instrução – por meio do Relógio (*Clock*).

É o dispositivo mais complexo da UCP. Ele possui a lógica necessária para realizar a movimentação de dados e de instruções de e para a UCP, através dos sinais de controle que emite em instantes de tempo programados. A **figura** mostra um diagrama em bloco simplificado dos principais elementos da função controle de um processador, ressaltando a Unidade de Controle, UC. Como também se observa na **figura**, a UC se conecta a todos os principais elementos do processador e ao barramento de controle, como, por exemplo, a UAL. Os sinais de controle, emitidos pela UC, ocorrem em vários instantes durante o período de realização de um ciclo de instrução e, de modo geral, todos possuem uma duração fixa e igual, originada em um gerador de sinais denominado relógio (*clock*).

O Relógio é responsável por definir esse tempo de duração fixa e igual na geração dos sinais de controle.

A arquiteturas da Unidade de Controle foi projetada para executar microoperações (menor parte individual executável pelo processador) por meio de **Micro programação** (pela execução de uma micro instrução) que é a programação prévia diretamente no hardware. Cada micro operação é realizada por iniciativa de um pulso originado na UC.

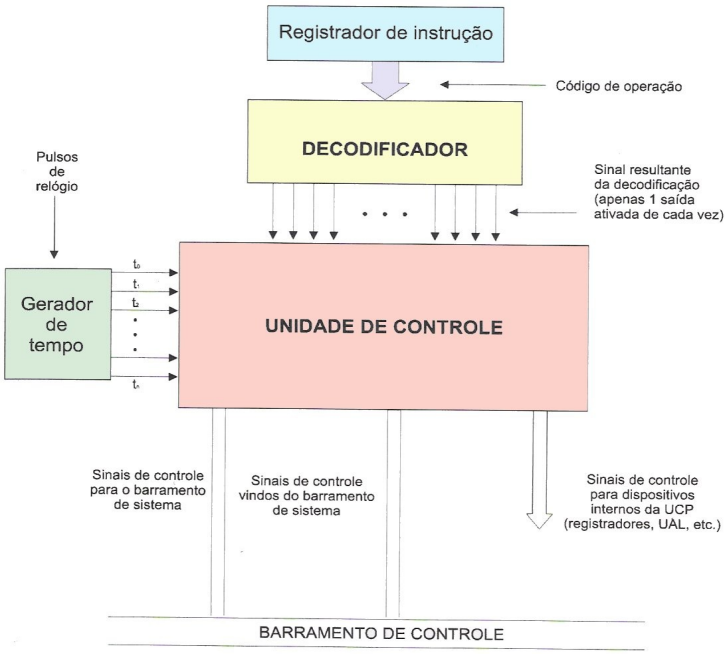


Diagrama em bloco simplificado da função controle

Assim, por exemplo, o início de um ciclo de instrução consiste em buscar (*fetch*) a referida instrução e trazer uma cópia sua da MP para o processador (para o registrador de instrução, como será explicado mais adiante). Para efetivar esta ação são realizadas algumas ações menores que, em conjunto, constituem a desejada transferência (na realidade, constituem os passos de um ciclo de leitura, conforme descrito **anteriormente**). Tais operações menores denominam-se microoperações, por se constituírem na menor parte individualmente executável pelo processador. A **figura** mostra as microoperações realizadas para completar o referido ciclo de busca.

Cada microoperação é realizada por iniciativa de um pulso originado na UC em decorrência de uma prévia programação (diretamente no hardware ou pela execução de uma microinstrução, se a arquitetura do processador é microprogramada).

Exemplo de uma sequência de micro operação para o ciclo de busca (fetch cicle):

Tempo	Micro operação	Descrição
t0	**REM <== (CI)**	O REM recebe o conteúdo do Contador de Instrução referente ao endereço da próxima instrução a ser executada pelo processador.
t1	**CI <== CI + N** **RDM <== M(op.)**	O Contador de Instrução é atualizado com o endereço da próxima instrução a ser realizada. O RDM é carregado com a micro operação a ser executada, oriunda da Memória Principal.
t2	**RI <== RDM**	O Registrador de Instrução é carregado com a micro operação oriunda do RDM.
t0, t1 e t2 => pulsos de relógio emitidos sequencialmente		

M(op.) => conteúdos de células contendo micro operação

Arquitetura de processadores

Modo como são executadas as instruções de máquina pelo processador, em particular como a Unidade de Controle gerencia o processo. Veremos como as duas primeiras arquiteturas se constituem, mas as outras duas não serão abordadas:

Single Instruction Stream Single Data Stream (SISD) – execução sequencial das instruções. É o modelo arquitetural de von Neumann.

Pipeline – execução concorrente de processos.

Processamento paralelo

Processamento vetorial

Relógio – Clock

É o dispositivo gerador de pulsos cuja duração é chamada de ciclo.

Frequência: quantidade de vezes por segundo que o ciclo se repete

Ciclo de relógio ou de máquina (***machine cycle***): é o intervalo de tempo entre o início de um pulso e o início do seguinte

Micro operação: são subciclos que acionam pas-sos diferentes da operação elementar

É o dispositivo gerador de pulsos cuja duração é chamada de ciclo. A quantidade de vezes em que este pulso básico se repete em um segundo define a unidade de medida do relógio, denominada *freqüência*, a qual também usamos para definir velocidade na UCP. Um ciclo de relógio ou de máquina (*machine cycle*) é o intervalo de tempo entre o início de um pulso e o início do seguinte. Este ciclo está relacionado à realização de uma operação elementar, durante o ciclo de uma instrução. No entanto, mesmo esta operação elementar não se realiza em um só passo e, por essa razão, costuma-se dividir o ciclo de máquina em ciclos menores (subciclos), defasados no tempo, de modo que cada um aciona um passo diferente da operação elementar. Esses diferentes passos de uma operação elementar denominam-se microoperações, conforme já mencionamos anteriormente.

Na **figura** apresentamos um exemplo de microoperações realizadas para completar a busca de uma instrução da MP para o Registrador de Instrução, RI, na UCP. Cada microoperação é realizada em um instante de tempo T_n, conforme se observa na figura. Esses instantes de tempo são originados no relógio.

Se as operações, para realizar um ciclo de instrução, duram o tempo definido por um ou mais pulsos do relógio, e se estes pulsos tiverem curta duração, mais operações podem ser realizadas na mesma unidade de tempo (o período-base utilizado é o segundo, ciclos por segundo, milhões de instruções por segundo, etc.).

A unidade de medida utilizada para a freqüência dos relógios de UCP é o hertz (Hz), que significa 1 ciclo por segundo — mesma unidade de medida de freqüência de sinais analógicos, como a voz. Como se trata de freqüências elevadas, abreviam-se os valores usando-se milhões de hertz, ou de ciclos por segundo (megahertz, ou simplesmente MHz).

Frequência

A unidade é o Hz (ciclo por segundo). Um processador que funciona com seu relógio oscilando 25 milhões de vezes por segundo → frequência de 25 Mhz.

A duração de um ciclo (T) vale: 1/25.000.000 = 0,00000004 segundos ou 40 nanosegundos ou 4 x 10 ^ -9

Qual o tempo T para um processador de 2 GHz?

A frequência é um indicador de desempenho dos processadores. Não significa que processadores com maior clock são mais eficientes que os de menor clock – Deve-se considerar a tecnologia e arquitetura de construção dos mesmos.

Se as operações, para realizar um ciclo de instrução, duram o tempo definido por um ou mais pulsos do relógio, e se estes pulsos tiverem curta duração, mais operações podem ser realizadas na mesma unidade de tempo (o período-base utilizado é o segundo, ciclos por segundo, milhões de instruções por segundo, etc.).

A unidade de medida utilizada para a freqüência dos relógios de UCP é o hertz (Hz), que significa 1 ciclo por segundo — mesma unidade de medida de freqüência de sinais analógicos, como a voz. Como se trata de freqüências elevadas, abreviam-se os valores usando-se milhões de hertz, ou de ciclos por segundo (megahertz, ou simplesmente MHz).

Registrador de Instrução (R I)

Tem a função de armazenar a instrução a ser executada pela CPU.

Cada ciclo de instrução realiza um ciclo de busca (acesso à memória) e ao término deste ciclo de leitura a instrução é armazenada no Registrador de Instrução (RI), via barramento de dados e RDM.

É o registrador (mostrado na **figura**) que tem a função específica de armazenar a instrução a ser executada pela UCP. Ao se iniciar um ciclo de instrução, a UC emite sinais de controle em seqüência no tempo, de modo que se processe a realização de um ciclo de leitura para buscar a instrução na memória (uma cópia dela). Conforme definido na programação do ciclo de busca de um ciclo de instrução, ao término deste ciclo de leitura a instrução desejada será armazenada no RI, via barramento de dados e RDM (no item 6.4 são apresentados mais detalhes da execução completa de um ciclo de instrução).

Contador de Instrução (C I) – Program Count (P C)

Armazena o endereço da próxima instrução a ser executada.

Após a busca da instrução a ser executada, o sistema executa automaticamente a modificação do CI. Na verdade o CI, em um ciclo de instrução, já contém o endereço da próxima instrução. É o registrador crucial para o processo de controle e de sequenciamento da execução dos programas.

É o registrador cuja função específica é armazenar o endereço da próxima instrução a ser executada. Tão logo a instrução que vai ser executada seja buscada (lida) da memória para a UCP (início do ciclo de instrução), o sistema automaticamente efetiva a modificação do conteúdo do CI de modo que ele passe a armazenar o endereço da próxima instrução na seqüência. Por isso, é comum definir a função do CI como a de "armazenar o endereço da próxima instrução"; na realidade, durante a maior parte da realização de um ciclo de instrução, o CI contém o endereço já da próxima instrução.

O CI é um registrador crucial para o processo de controle e de seqüenciamento da execução dos programas.

Decodificador de Instrução

Identifica que operação será executada. Cada instrução é uma ordem à CPU para executar determinada operação. Como são muitas instruções – cada uma tem um código próprio. Cabe ao decodificador identificar que instrução será realizada e assinalar à UC. Para cada instrução ele emite sinais em saídas

distintas para a UC.

É um dispositivo utilizado para identificar que operação será realizada, correlacionada à instrução cujo código de operação foi decodificado. Em outras palavras, cada instrução é uma ordem para que a UCP realize uma determinada operação . Como são muitas instruções, é necessário que cada uma possua uma identificação própria e única. A unidade de controle está, por sua vez, preparada para sinalizar adequadamente aos diversos dispositivos da UCP, conforme ela tenha identificado a instrução a ser executada.

REM e RDM

Este registradores já foram falados por ocasião do assunto de subsistemas de memórias.

5.2 Instruções de Máquina

O que é uma Instrução de Máquina?

São instruções detalhadas em pequenas etapas e implementadas no hardware (projetado para esse fim). É a formalização de uma operação básica (primitiva) que o hardware é capaz de realizar diretamente.

A arquitetura básica dos processadores do tipo von Neumann (**visto antes**), cujos princípios fundamentais ainda são válidos (a seqüencialidade da execução das instruções deixou de existir com o advento das técnicas *pipelining*, ou de processamento paralelo, ou vetorial, etc.), é essencialmente calcada na existência de uma ordem ou instrução para que o processador (o hardware) realize uma determinada operação (assim como nós instruímos um funcionário a realizar uma determinada atividade — a instrução, aqui, não no sentido de ensinar, mas de comandar a realização de um ato).

Ex: pode-se fabricar uma CPU com uma UAL capaz de somar ou de multiplicar dois números, mas não a seguinte operação matemática:

X = A + B*C (de uma só vez)

Tal operação, a UAL tem que ser instruída para executar:

T=B*C; e em seguida: X = A + T

Uma máquina pode executar tarefas complicadas e sucessivas se for "instruída" sobre o que fazer e em que seqüência isso deve ser feito. Os seres humanos (pelo menos a maioria), ao receberem uma instrução do tipo "trazer a pasta da funcionária Maria", são capazes de localizar o arquivo em que as pastas de todos os funcionários estão arquivadas — em geral por ordem alfabética — e achar a pasta, trazendo-a a quem pediu. Nosso cérebro realizou uma série de ações intermediárias para que a tarefa fosse concluída com êxito.

No entanto, se a mesma "instrução" fosse dada a uma máquina (e ela não tivesse qualquer outra orientação prévia armazenada), ela não conseguiria "trazer a pasta desejada".

Para a máquina, é necessário que a "instrução" seja detalhada em pequenas etapas, visto que ela é construída para ser capaz de entender só dessa forma, ou seja, em pequenas operações.

No exemplo em questão, a máquina deveria receber um conjunto de instruções como o exemplo abaixo, específicas para ela, sendo, portanto, chamadas de instruções de máquina (da máquina).

Exemplo de Instruções primitivas (localizar e buscar pasta)

achar arquivo. Se não houver arquivo, vá p/9; senão, prosseguir

comparar n° arquivo com n° arquivo que contém as pastas dos funcionários

se n° iguais, então prosseguir; senão, voltar para 1

achar uma pasta. Se não houver mais pastas, vá p/9; senão, prosseguir

comparar nome da pasta com nome dado

se forem iguais, então prosseguir; senão, voltar para 4

retirar a pasta

abrir a pasta para quem pedir

parar

Exemplo de um Conjunto de Instruções:

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

O projeto de um processador é centrado no conjunto de instruções de máquina que se deseja que ele execute. A análise e decisão do projeto envolve:

O **tamanho** das instruções.

A **complexidade** das instruções.

Obs: quanto menor e mais simples o conjunto de instruções, mais rápido é o ciclo de tempo do processador.

Tecnologias para mini, microcomputadores e Estações de Trabalho:

CISC – Sistemas com conjunto de instruções complexo

RISC – Sistemas com conjunto de instruções reduzido

Um projeto de processador resume-se em:

1) Definir o conjunto de instruções:

- ✔ **Qual o formato**
- ✔ **Qual o tamanho de cada uma**
- ✔ **Quantidade de operações que irá realizar**

2) Implementar os componentes do processador em função do conjunto de instruções definido:

- ✔ **ULA**
- ✔ **Registradores**
- ✔ **Barramentos**
- ✔ **Decodificador de instrução, etc.**

Do ponto de vista físico (hardware):

Uma instrução de máquina é um grupo de bits que indica à CPU uma operação. Cada CPU é projetada para realizar um certo número de operações primitivas. Cada operação está associada a uma instrução de máquina.

A CPU tem instruções que realizam as operações:

- ✔ **operações matemáticas (aritmética, lógicas, deslocamento)**
- ✔ **movimentação de dados (memória – CPU e vice-versa)**
- ✔ **entrada e saída (R/W em dispositivos de E/S**
- ✔ **controle (desvio de sequência de execução, parar, etc.)**

Conjunto de Instruções

São todas as possíveis instruções de serem interpretadas e executadas por um processador. Exemplos:

- ✔ Intel 8080 => 78 instruções de máquina
- ✔ Intel 8088 => 117 instruções de máquina
- ✔ 80486 => 286 instruções de máquina
- ✔ Pentium II => 217 instruções de máquina

Temos assim dois conceitos bem distintos:

Instrução de máquina => instruções binária ou Assembly

Comando => instrução de linguagem de alto nível (C,C++, Pascal Java, etc.)

O projeto de um processador é centrado no conjunto de instruções de máquina que se deseja que ele execute (na realidade, do conjunto de operações primitivas que ele poderá executar). Uma das mais fundamentais análises e decisões do projeto envolve o tamanho e a complexidade do conjunto de instruções. Quanto menor e mais simples o conjunto de instruções, mais rápido é o ciclo de tempo do processador.

Atualmente, há duas tecnologias de projeto de processadores empregadas pelos fabricantes de mini, microcomputadores e de estações de trabalho:

- *Sistemas com conjunto de instruções complexo* (complex instruction set computers — CISC) e
- *Sistemas com conjunto de instruções reduzido* (reduced instruction set computers — RISC).

Na realidade, todo o projeto do processador se resume em:

- definir o conjunto de instruções (qual o formato e tamanho de cada uma, quais as operações a realizar — o que caracteriza a quantidade);
- implementar os componentes do processador em função da definição anterior (UAL, registradores, barramentos, etc.).

Do ponto de vista físico (do ponto de vista do hardware), uma instrução de máquina é um grupo de bits que indica ao processador uma operação ou ação que ele deve realizar. Um processador é fabricado com a capacidade de realizar uma certa quantidade de operações bem simples (primitivas), cada uma delas associada a uma instrução de máquina.

Formato das Instruções

De um modo geral, o grupo de bits da instrução são divididos em duas partes: **Código de Operação** e campos **Operandos**.

De modo geral, podemos separar o grupo de bits que constitui a instrução em duas partes: uma delas indica o que é a instrução e como será executada e a outra parte se refere ao(s) dado(s) que será(ão) manipulado(s) na operação. A primeira parte é constituída de um só campo, enquanto a segunda parte poderá ter um ou mais campos, conforme a instrução se refira explicitamente a um ou mais dados. Assim, temos os seguintes campos em cada instrução:

- um campo (um subgrupo de bits) chama-se *código de operação*;
- o restante grupo de bits (se houver) denomina-se *campo do(s) operando(s)* ou, simplesmente, operando(s).

Código de operação — C.Op. — é o campo da instrução cujo valor binário é a identificação (ou código) da operação a ser realizada. Assim, cada instrução possui um único código, o qual servirá de entrada no decodificador da área de controle. A tabela abaixo apresenta exemplos de tipos de operações primitivas normalmente encontradas na implementação dos processadores.

Um processador que possua instruções cujo campo C.Op. tenha uma largura de 8 bits poderá ser fabricado contendo a implementação de um conjunto de até 256 instruções diferentes, visto que:

C.Op. = 8 bits. Então: $2^8 = 256$ códigos de operação.

Como cada C.Op. representa uma única instrução, então 256 C.Op. indica 256 instruções de máquina.

Campo operando — Op. — é(são) o(s) campo(s) da instrução cujo valor binário indica a localização do dado (ou dados) que será(ão) manipulado(s) durante a realização da operação.

* Transferir uma palavra de dados de uma célula para outra.
* Efetuar a soma entre dois operandos, guardando o resultado em um deles ou em um terceiro operando.
* Desviar incondicionalmente para outro endereço fora da seqüência.
* Testar uma condição. Se teste verdadeiro, então desviar para outro endereço fora da seqüência.
* Realizar uma operação lógica AND entre dois valores.
* Parar a execução de um programa.
* Adicionar 1 ao valor de um operando.
* Transferir um byte de dados de uma porta de E/S para a MP.
* Transferir um byte de dados da MP para uma porta de E/S.
* Substituir o operando por seu valor absoluto.

Exemplo de operações primitivas típicas.

Código de Operação (C. Op.):

Seu valor binário identifica a operação a ser realizada e define como será executada a instrução. É constituído de um só campo (subgrupo de bits da instrução).

Um C.Op de largura de 8 bits é capaz de se implementar até 256 instruções, pois $2^8 = 256$. Há 2 maneiras de se criar um conjunto de instruções:

- ✔ **C.Op. de tamanho fixo e**
- ✔ **C.Op. de tamanho variável**

C. Op. de tamanho Fixo (características):

São mais simples de se implementar e mais simples de se manipular durante a execução. Em sistemas com grande quantidade de instruções tendem a crescer aumentando o tamanho da instrução e consequentemente requerem mais memória para armazenar os programas, um valor típico de seu tamanho é 8 bits.

Exemplos: Intel 8080 e 8085

C. Op. de tamanho variável (características):

Permitem codificar uma quantidade maior de instruções com menor quantidade de bits, onde há uma fixação do tamanho dos campos operandos (redução da quantidade de endereçamento de memória) ou necessidade de aumentar o tamanho da instrução.

Permite maior versatilidade entre a quantidade de bits do C.Op. e a quantidade de bits do(s) Operando(s), sem aumentar demasiadamente o tamanho total das instruções. O mais comum é utilizar conjuntos de instruções com C.Op. variável e instruções de tamanho total variável.

Campos Operando (Op.):

Pode ser um ou vários campos e é/são valores binários que indica(m) a localização do(s) dado(s) a serem utilizados durante a realização da operação.

Exemplo estrutural da operação "SOMAR"

Sabemos que a instrução possui duas partes: o C.Op. (código de identificação da operação) e os campos operandos. Como trata-se de soma, teremos:

2 operando que indicam os endereços dos 2 valores a serem somados

1 operando para indicar qual endereço que será armazenado o resultado

Poderia ser adotado apenas 2 operandos e explicitar que o resultado da operação será armazenado no Op1 ou Op2

Exemplo de formatos da instrução "SOMAR"

C. Op	Op.1	Op.2	Op.3	(3 operandos)

Op.3 <--- Op.1 + Op.2 (operando 3 recebe a soma do operando 1 e 2)

C. Op	Op.1	Op.2	(2 operandos)

Op.1 <--- Op.1 + Op.2 (operando 1 recebe a soma do operando 1 e 2)

ou

Op.2 <--- Op.1 + Op.2 (operando 2 recebe a soma do operando 1 e 2)

Poderia utilizar-se do registrador ACC para armazenar o 1° valor e a instrução teria o seguinte formato:

C. Op	Operando	(1 operando)

ACC <--- ACC + Operando (ACC recebe a soma do ACC e operando)

C.Op.	Operando 1	Operando 2	Operando 3

Utilizando-se uma forma mais específica para representar a operação que a instrução indica, teríamos:

(operando 3) ← (operando 1) + (operando 2)

A mesma operação (necessitando da mesma forma de três operandos) poderia ser realizada com outro tipo de instrução:

C.Op.	Operando 1	Operando 2

Neste último exemplo, a instrução pode indicar (pelo C.Op. específico e diferente do C.Op. da instrução anterior) que se deve somar o valor indicado pelo operando 1 com o valor indicado pelo operando 2, e que o resultado deve ser armazenado no operando 1 (poderia também ser no operando 2).

A instrução seria assim representada:

(operando 1) ← (operando 1) + (operando 2)

Ou ainda:

(operando 2) ← (operando 1) + (operando 2)

Considerações sobre Formatos de Instrução

Em um mesmo conjunto de instruções de um processador podem existir formatos diferentes de instruções. Os exemplos anteriores são parte de formatos possíveis, existem outros. O significado do valor binário dos operandos, ou seja, modo de localizar o dado, varia de instrução para instrução a isso chamamos de **modo de endereçamento**.

Pode-se observar, então, que, em um mesmo conjunto de instruções de um processador, podem existir formatos diferentes de instruções, inclusive para a realização de uma mesma operação. Os formatos apresentados são uma parte da quantidade de formatos que podem existir distribuídos em processadores reais.

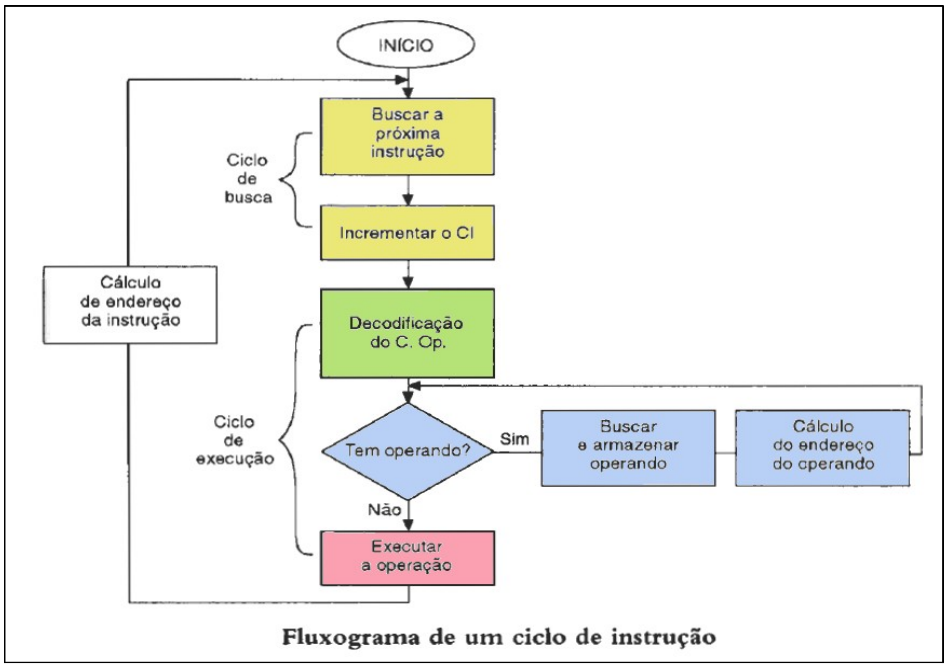
Um outro fator a ser considerado no projeto do conjunto de instruções de um processador refere-se ao significado do valor binário indicado no(s) campo(s) operando(s) das instruções. Ou seja, o modo de localizar o dado pode variar de instrução para instrução. Chama-se a isto de **modo de endereçamento**; atualmente há vários destes modos sendo empregados nos processadores. Para evitar o risco de tornar a explicação inicial do assunto mais complicada do que o desejado, apresentaremos apenas instruções com formato e características bem simples: apenas dois modos de endereçamento e somente 1 operando.

5.3 Ciclo de Instruções

Como visto na parte 1 de processadores, o ciclo de instrução executa 4 etapas distintas, marcadas em cores na figura.

- ✔ **Busca da instrução**
- ✔ **Interpretação da instrução**
- ✔ **Busca dos Dados**
- ✔ **Execução da instrução**

Após isso o ciclo se repete.



Fluxograma de um ciclo de instrução

5.4 Pipelining

Metodologia de Linha de Montagem - PipeLine

Criada por Henry Ford no Sec. XX em uso até hoje e consiste em dividir o

processo de execução das instruções em estágios independentes uns dos outros, ou seja, vários ciclos de instrução podem se superpor, diferentemente pelo adotado por John von Neumann. Entende-se por estágio cada etapa do ciclo de instrução.

Exemplo de Ciclos de dois estágios:

1º estágio: Leitura ou busca da instrução

2º estágio: Execução da instrução lida

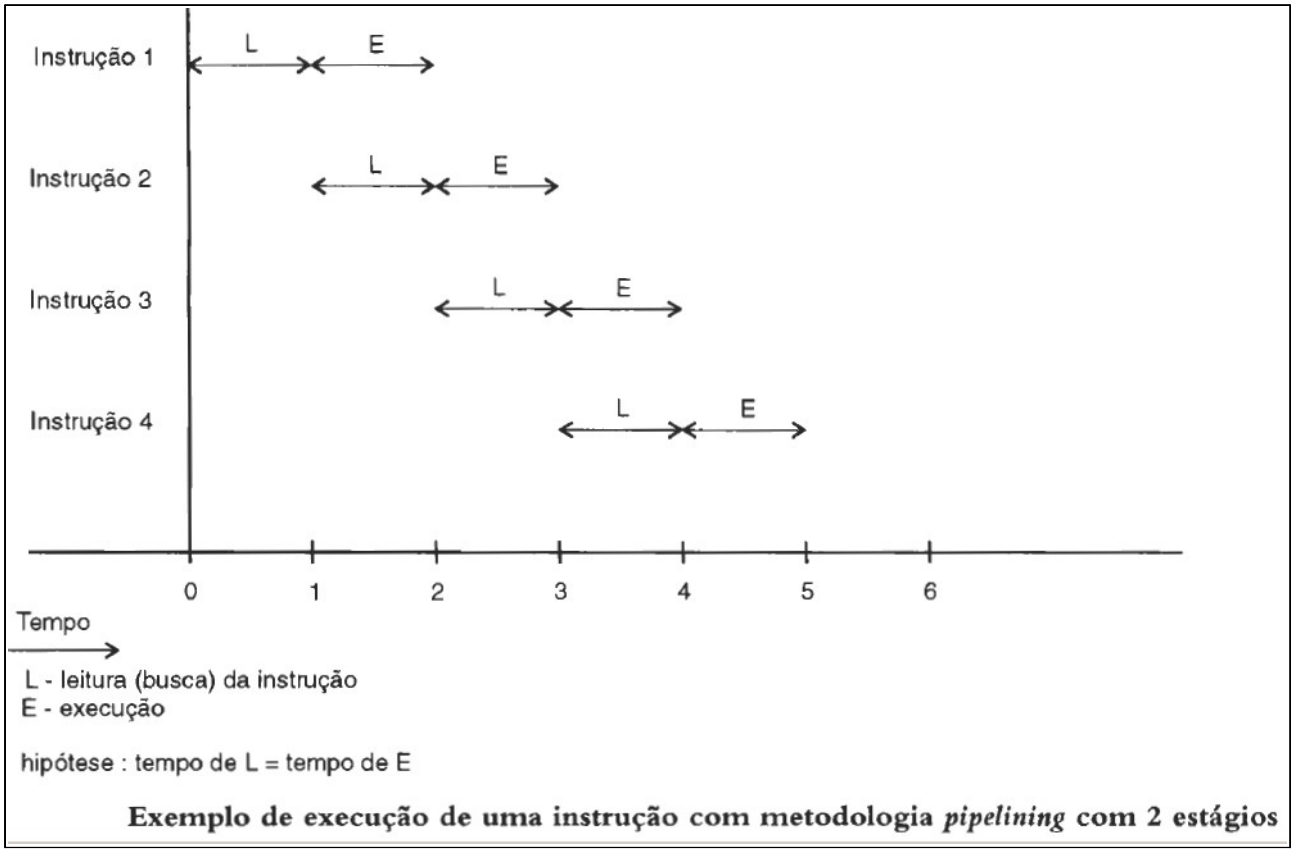
Ao descrever o funcionamento da UCP na realização de seus ciclos de instrução foi observado que, embora o ciclo de instrução seja composto de várias etapas (ver o fluxo da **figura**), ele é realizado basicamente de forma seqüencial, isto é, uma etapa se inicia após a conclusão da anterior. Este procedimento segue tipicamente o conceito de arquitetura de um sistema de computação proposto por von Neumann e que já foi definido como sendo do tipo SISD.

UCPs deste tipo vêm sendo usadas desde as primeiras gerações de computadores, e muitos aperfeiçoamentos tecnológicos foram introduzidos para reduzir o tempo de processamento de uma instrução, entre os quais o aumento da velocidade do relógio e a tecnologia de semicondutor, com seus sucessivos melhoramentos em fabricação e miniaturização. Porém, as etapas do ciclo de instrução permaneciam sendo realizadas seqüencialmente, como mostrado na **figura** . Na figura podemos observar este fato e verificar que o tempo total de execução da instrução é a soma dos tempos gastos em cada etapa.

O processo é semelhante ao da fabricação de um automóvel, caso ainda existisse uma fábrica que funcionasse em montagem seqüencial: primeiro, a carroceria (gastaria um tempo T_1), depois o motor (tempo T_2), em seguida as rodas (tempo T_3), depois a parte elétrica (T_4), os bancos e espelhos (T_5) e o acabamento final (T_6). Cada carro ficaria pronto em um tempo $T = T_1+T_2+T_3+T_4+T_5+T_6$ e somente com a total conclusão de um carro (tempo T) é que um outro automóvel iniciaria sua montagem. A fábrica produziria, em um turno de X horas, um total máximo de X/T carros, o que sempre seria um valor pequeno.

Suponhamos que o processo de realização do ciclo de uma instrução seja dividido em dois estágios: o da *leitura ou busca da instrução* e o *da execução da instrução* lida.

Neste caso, para ler uma instrução, é necessário um acesso à memória, mas para executar a instrução nem sempre é necessário acessar a memória (por exemplo, na decodificação e na execução da operação não há acessos à memória). Portanto, é possível ler uma instrução, utilizando-se dos circuitos de um estágio, e transferir esta instrução para o estágio de execução. E, durante o período em que, neste estágio, não há atividade com a memória, pode-se ativar o estágio de leitura para buscar uma nova instrução e continuar o processo com novas instruções, como mostrado na Fig. Esta atividade de busca de nova instrução antes da conclusão da anterior é conhecida como pré-busca (*pre-fetch*).



Exemplo de execução de uma instrução com metodologia *pipelining* com 2 estágios

T = tempo de execução da instrução = $T_1 + T_2$

Ciclo de busca = leitura da instrução, incremento do CI

Ciclo de execução = decodificação, busca do operando, execução da operação

Ciclo de instrução em execução seqüencial

Observa-se que o tempo de para executar duas instruções com pipeline consome o mesmo tempo que o utilizado para executar uma única instrução com arquitetura SISD de Neumann, portanto mais eficiente e por isso usada nos processadores atuais.

Na verdade os processadores implementam pipeline com cinco estágios ao invés de 2 como representado na figura acima. Isso possibilita mais eficiência e reduz o tempo de espera do processador quando estiver executando uma operação que não realiza busca de dados, por exemplo, como é o caso de uma operação de desvio.

5.5 Barramento síncrono e assíncrono

Barramentos: São as vias que interligam os diversos componentes do Sistema Computacional para trafegar os bits de dados e controle.

Largura ou tamanho do barramento => medida de desempenho do Sistema (taxa de transferência)

Barramento Síncrono: os pulsos emitidos pelo relógio sincronizam o funcionamento das diversas linhas do barramento – ciclo do relógio ou de barramento

Barramento Assíncrono: não há relógio sincronizador para funcionamento das linhas do barramento – cada evento depende do anterior e com duração de tempo diferente

Comparação:

O barramento síncrono é simples de implementar e testar por ser inflexível no tempo (atividades de mestre e escravo).

Desvantagem: incompatibilidade de lidar com barramento de velocidades diferentes (tecnologias antigas) – maior tempo de espera para sincronização.

Isso não ocorre com o barramento assíncrono – tecnologias usada nos sistemas atuais.

5.7 Overclocking

O funcionamento do computador é cadenciado pelos pulsos do relógio do processador, isto é, cada micro operação é realizada pelo estímulo desses pulsos. Quanto menor o intervalo entre esses pulsos mais rápido são realizadas essas micro operações.

Cada processador é projetado para funcionar em certa faixa de

velocidade (emissão de pulsos do clock), ou seja, o sistema pode ser configurado pelo seu SETUP para definir o valor dessa velocidade – dentro da capacidade projetada para isso.

Overclock – não existe uma tradução específica, é usada no inglês mesmo.

Vem a ser o aumento da frequência de operação do relógio do processador (acima do especificado pelo fabricante). O processador com overclock opera mais rápido, mas em compensação compromete o tempo de sua vida. Não é recomendado sua utilização (pode causar travamentos).

Se você que um processador mais veloz, compre um com maior frequência de operação.

EXERCÍCIOS

1) Descreva as funções básicas de uma UCP, indicando os seus componentes principais.
2) Quais são as funções da Unidade Aritmética e Lógica — UAL?
3) O que é e para que serve o ACC?
4) Qual é o componente de um processador que determina o período de duração de cada uma de suas atividades e controla o sincronismo entre elas?
5) Quais são as funções da unidade de controle de um processador?
6) Seria possível realizar o projeto de um processador no qual o tamanho em bits do CI fosse diferente do tamanho do REM? Nesse caso, qual dos dois registradores deveria ter maior tamanho? Por quê?
7) Considere um computador cuja MP é organizada com N células de 1 byte cada uma. As instruções interpretadas pela UCP possuem três tamanhos diferentes: as do tipo A possuem 16 bits; as do tipo B
8) Qual é e onde se localiza o registrador cujo conteúdo controla a seqüência de processamento das instruções de um programa?
9) Qual é o registrador cujo conteúdo determina a capacidade de memória de um computador? Justifique.
10) Um computador tem um REM de 16 bits e uma barra de dados de 20 bits. Possui instruções de um operando, todas do tamanho de uma célula de memória e de mesmo tamanho da palavra. Ele foi adquirido com apenas uma placa de 4K de memória.

    Pergunta-se:

    a) Qual o tamanho, em bits, do RDM e CI?

    b) Seria possível aumentar-se a capacidade de armazenamento dessa memória? Até quanto? Por quê?

    c) Qual a quantidade máxima de instruções de máquina que poderia existir nesse computador?

REFERÊNCIAS

MONTEIRO, M. A. Introdução a Organização de Computadores, 2004.

Sites da Internet.

Fim – Parte 2