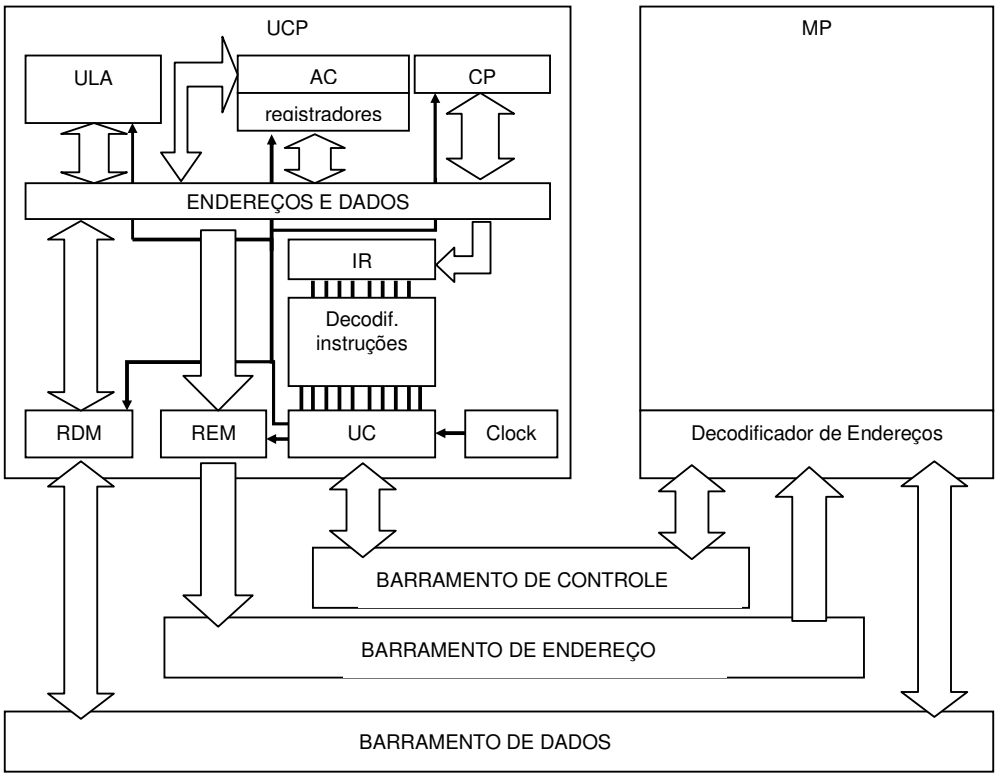
Tópico 06 – UNIDADE CENTRAL DE PROCESSAMENTO

Introdução: Iremos estudar com um pouco mais de aprofundamento a UCP, nosso foco é a descrição dos registradores, do decodificador de instruções da unidade de controle e aspectos referentes a linguagem de máquina.

6.1 – UCP – Unidade Central de Processamento:

Sua função é computar, processar, calcular. Em nossos estudos estamos sempre considerando um processador simplificado, que obedece a arquitetura SISD (execução de uma instrução para um único conjunto de dados/operandos). Veja o esquema a seguir:



Funções Básicas

- Operações Internas (funções de processamento)
- Funções de Controle (sinais de controle p/ outros componentes do computador

Operações Primitivas

São instruções de máquina (binárias) que identificam a operação a ser executada

- Somar, subtrair, etc.
- Mover dados

Ex.: Soma

Código binário que identifica a operação de soma	Endereço dos operandos

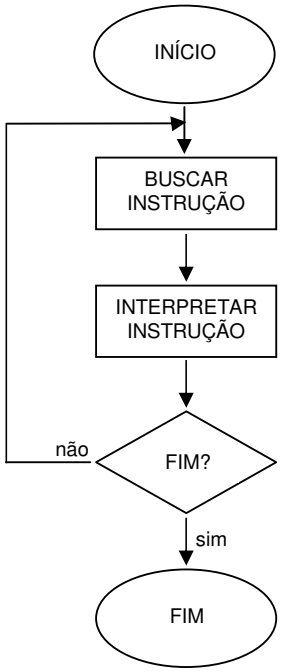
Programa Executável
Conjunto seqüencial de instruções de máquina. Para executá-lo é necessário:

- Carregá-lo na Memória Principal (ou carregar parte dele)
- Colocar o endereço do início do programa no registrador contador de programa (CP/PC – aponta o endereço das instruções)

Ciclo de Instrução
São os passos necessários para a execução das instruções de um programa pela UCP. Ex.:

1. Buscar a instrução (leitura)
2. Interpretar a instrução (tradução)
3. Buscar os dados, se necessário (carga dos dados)
4. Executar a instrução
5. Reiniciar o ciclo buscando a próxima instrução, se existir.

Fluxograma (de 1 a 5):



Pipeline
A fim de aumentar a velocidade de processamento, muitas estratégias de processamento foram implementadas, entre elas a chamada de "pipeline" onde o processador poderá iniciar um ciclo de busca por uma nova instrução enquanto executa a operação com a instrução atual (semelhante a uma linha de produção de uma fábrica onde os setores vão executando suas operações separadamente e todos ao mesmo tempo). Os processadores atuais podem ter 10 estágios de pipeline (até 10 ações diferentes podem estar sendo executadas pelo processador, sem que cada ação atrapalhe a execução da outra).

6.2 – Registradores
São as unidades internas de memória utilizadas em diversas operações. Podem ter tamanhos diferentes e podem ser separados basicamente em:

- Registradores de propósito geral e
- Registradores especiais

Ex.: Em algumas arquiteturas poderíamos ter o seguinte conjunto de registradores:

- Acumulador (ACC)
- Registrador(es) temporário(s)
- Flags

- Registradores de propósito geral

  E exemplo mostra uma arquitetura de registradores semelhante aos antigos processadores 80xx onde temos um registrador acumulador, que recebe operando e o sobrescreve com o resultado da operação (caso seja operação aritmética). Registradores temporários e de propósito geral, que auxiliam na execução das operações lógicas e aritméticas e registradores especiais chamados flags (bandeiras), que são utilizados em diversas sinalizações dos estados dos componentes internos (e externos) a UCP.

Em arquiteturas mais atuais podemos ter registradores específicos como: sinal, overflow, zero, carry (vai um), paridade, etc.

Contador do Programa (CP/PC) ou Contador de Instrução (CI/IC)
Este registrador possui função de grande importância pois é ele que recebe o endereço do início de qualquer programa (ou bloco) que esteja na MP. Uma vez carregado com o endereço inicial, este será incrementado a fim de percorrer todas as linhas do programa (ou bloco). Nos sistemas atuais cada bloco dos diferentes programas em execução terá seu endereço de início que será incrementado de acordo com o avanço da execução do mesmo independentemente dos demais programas.

Observações sobre o Tamanho da Palavra
Originalmente a idéia que se tem é que o tamanho dos registradores (ou a palavra) tem a mesma quantidade de bits que o barramento de dados/endereços (quantidade de linhas no barramento). Isto foi verdade para arquiteturas de UCPs como as 80xx citadas anteriormente. Com o avanço da tecnologia e o advento das cachê L1, pode-se aumentar o tamanho do barramento sem necessariamente aumentar o tamanho da palavra. Como exemplo podemos citar arquiteturas onde a palavra (registradores de propósito geral, etc) possui 64 bits enquanto que o barramento interno possui 128 bits até a cachê L1.

6.3 – Unidade de Controle
A unidade de controle é responsável pelos sinais de controle (ativações, habilitações, desabilitações) tanto de dispositivos internos quanto externos a UCP. Estas ações devem ser organizadas e sincronizadas, pois as velocidades envolvidas são muito altas e os dispositivos eletrônicos possuem características elétricas que devem ser consideradas em sua ativação/desativação.

Registrador de Instrução (RI/IR)
Este registrador recebe o código da instrução a ser executada via barramento de dados/endereços (ver figura no início deste roteiro) e o transfere ao decodificador de instruções. A quantidade de bits desse registrador determina a quantidade de instruções possíveis.

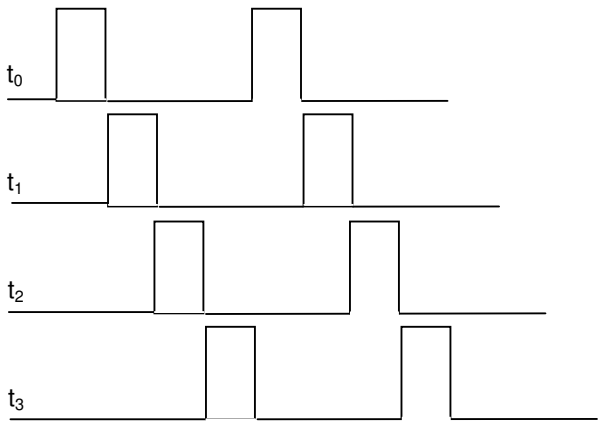
Decodificador de instruções
Este dispositivo possui barramento de comunicação com o RI. Ao receber um código de uma instrução, traduz o mesmo e gera o sinal específico relacionado à instrução. O Decodificador de Instruções terá o número de linhas de sinal igual a quantidade de instruções possíveis.

> Ex.: Se um sistema possui RI com 8 bits, poderemos ter 2^8=256 instruções diferentes e igual número de linhas saindo do decodificador e chegando a Unidade de Controle.

Relógio (clock) e Contador de Tempo
O circuito da Unidade de Controle deve controlar e sincronizar todas as operações a serem executadas, para que isso seja possível a própria unidade deve trabalhar de forma sincronizada. Para possibilitar esta ação seu circuito eletrônico funciona com pulsos elétricos com intervalo de duração constante. Na verdade praticamente todos os sistemas dentro do computador deverão trabalhar sincronizados.
Como as instruções por sua vez são separadas micro-instruções, faz-se necessário criar um mecanismo que “quebre” o pulso de clock em pulsos menores. Para executar essa ação existe um dispositivo chamado contador de tempo que recebe o pulso de clock e o solta para o sistema em linhas separadas. Cada uma dessas linhas terá o pulso do clock com atraso igual a largura do pulso de clock original, possibilitando criar a temporização para as micro-instruções.

Ex.: No desenho a seguir estão representados 4 sinais:

- t_0 – sinal de clock original
- t_1 a t_3 – sinais gerados pelo circuito contador de tempo



6.4 – Instruções de Máquina e Linguagem Assembly

Podemos entender uma instrução como sendo formada por uma palavra binária descrita em uma arquitetura específica. Essa arquitetura deve ter um conjunto de instruções reconhecidas pelo decodificador de instruções.

A passagem de instruções para a UCP deve ser feita em binário (linguagem de máquina), o que se torna cansativo e complexo. A fim de facilitar o trabalho criou-se uma linguagem que converte códigos de mais fácil entendimento para os seres humanos. Essa linguagem se chama Assembly (montagem) e o programa que converte (compila) os comandos do assembly para a linguagem de máquina é chamado Assembler (montador).

Existem diversas versões de assembly para as diversas arquiteturas de UCP.

Ex.: A seguir demonstra-se uma expressão descrita em linguagem C, seu equivalente em binário, hexadecimal e em uma versão da linguagem assembly:

- **Código em linguagem C:** X = A + B – C;

- **Código em hexadecimal:**
  123H
  324H
  425H
  226H

- **Código em binário:**
  0001 0010 0011
  0011 0010 0100
  0100 0010 0101
  0010 0010 0110

- **Código em assembly:**

LDA A	(load Acc)	Acc ← A
ADD B	(Acc add B)	Acc ← Acc + B
SUB C	(Acc sub C)	Acc ← Acc - C
STR X	(store X)	Acc → X

Cada um dos códigos em Assembly (também chamados de OpCods ou códigos operacionais) é criado de forma “mnemônica”, ou seja, o código se parece com a palavra que descreve a operação que ele realiza (na língua inglesa). Assim:

- LDA – Carregar no acumulador o conteúdo do reg. especificado
- ADD – Somar o conteúdo do acumulador com o conteúdo do reg. especificado
- SUB – Subtrair o conteúdo do acumulador com o conteúdo do reg. especificado
- STR – Colocar o conteúdo do acumulador no reg. especificado