Unidade central de Processamento (Processador)

- Realiza operações básicas codificadas em 0s e 1s
- Instrução contém código de operação e informação dos operandos
- Programa é composto de instruções de máquina armazenadas em seqüência na memória

Ciclo de instrução

1. Busca a próxima instrução (apenas uma)
2. Interpreta a instrução (decodificação)
3. Buscar os dados
4. Executa a instrução guardando o resultado
5. Volta para passo 1

Atividades da UCP

- Função de Processamento:
    - Atividades relacionadas a execução (processar)
    - ULA (Unidade Lógica e Aritmética) e registradores
- Função de Controle:
    - Atividades de busca, interpretação e controle da execução e dos demais componentes (E/S, memória)
    - PC (Contador de Instruções), Unidade de controle e relógio

Instrução de máquina

- Operações básicas que podem ser executadas pelo hardware:
    - somar dois números
    - multiplicar 2 números
    - transferir palavra de dados da memória para UCP
    - desviar s seqüência de execução (condicionalmente e incondicionalmente: sem condição)
    - operações lógicas
    - transferir bytes da E/S para memória principal
- CISC (complex instruction set computer): conjunto de instruções complexas para aumentar a eficiência
- RISC (reduced instruction set computer): conjunto limitado e simples de instruções e número grande de registradores.
- Projeto de um processador inclui definir o conjunto de instruções e implementar o hardware para executá-lo
- Instruções podem ter tamanho fixo ou variável

Linguagem de montagem (Assembly)

Ex: *Programa em C*

```
a=b+c
```

Programa em Linguagem de montagem:

```
add b c a
```

add: operação: identifica a instrução
b, c, a: operando(s): símbolos que representam o endereço de memória ou registradores utilizados para armazenar os dados referidos pela instrução.

Ex: *Programa em C:*

```
f=(g+h)-(i+j)
```

Programa em Linguagem de montagem:

```
add g h t0 (soma de g com h é colocada em t0)
add i j t1 (soma de i com j é colocada em t1)
sub t0 t1 f (subtrai de t0 com t1 é colocada em f)
```

Variáveis ficam em registradores: 8 registradores de 32 bits cada um: 0,1,2,3,4,5,6 e 7
Supondo g alocada para registrador 1, h alocada para registrador 2, i alocada para registrador 3, j alocada para registrador 4 e f alocada para registrador 5:

```
add $1 $2 $6
add $3 $4 $7
sub $6 $7 $5
```

Ex: *Programa em C*

```
g=h+A[8]
```

Programa em linguagem de montagem:
Estruturas complexas como arrays podem conter mais dados que registradores disponíveis para armazená-los. Então a estrutura é mantida na memória. Só as operações lógicas e aritméticas são realizadas entre registradores. Logo, necessita-se de instruções para realizar transferência entre a memória e registradores.

Supondo g alocada no registrador 2, h alocada em 3 e endereço inicial de A se encontra no registrador 4:

```
lw $4 $1 8   (lw: load word: carrega o conteúdo da memória para o
               registrador, isto é, grava em $1 o valor que está na
               posição 8 do vetor A cujo endereço inicial é $4)
add $1 $3 $2
```

Ex: *Programa em C*

```
A[12]=h+A[8]
```

Programa em linguagem de montagem, supondo h alocada no registrador 2 e endereço inicial de A no registrador 4:

```
lw $4 $1 8     (coloca em $1 o conteúdo de $4 na posição 8)
add $1 $2 $1   (adiciona h com A[8])
```

```
sw $4 $1 12     (sw: store word: guarda na posição 12 do vetor com
                   endereço inicial em $4 o valor do registrador $1)
```

lw e sw são chamadas de instruções de transferência de dados. É necessário saber o *endereço* para acessar a palavra (vetor) na memória.