# Unidade de Controle

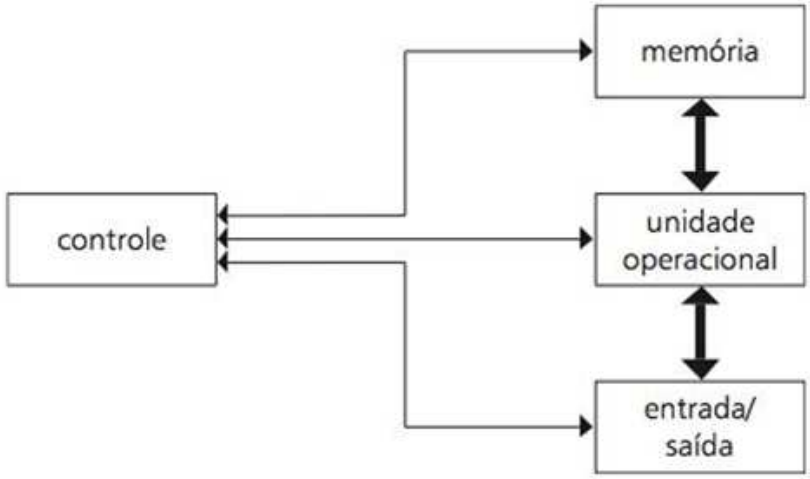
Prof. Alexandre Beletti

(Cap. 3 – Weber, Cap.8 – Monteiro,
Cap. 10,11 – Stallings)

# UC - Introdução

- Para gerenciar o fluxo interno de dados e o instante em que ocorrem as transferências entre uma unidade e a outra são necessários SINAIS DE CONTROLE, que são fornecidos pela UC.
- Unidade operacional e de controle estão juntas e reunidade na UCP

# UC – ativar/habilitar componentes




# Micro-Operação

- Cada sinal comanda uma micro-operação, que pode ser:
  - Uma carga em um registrador
  - Seleção de um dados para entrada em um determinado componente
  - Ativação da memória
  - Seleção de uma operação da ULA
  - Habilitação de um circuito lógico

# Organização da UC

- Convencional
- Microprogramada

# Convencional

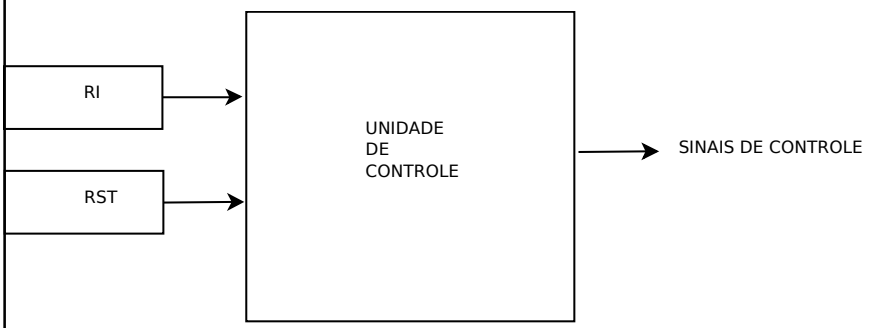
- A unidade de controle é formada por componentes digitais, como flip-flops, contadores e decodificadores, que geram, sequencialmente e nos instantes de tempo adequados, todos os sinais de controle necessários à ativação da unidade operacional, do sistema de entrada e saída e da memória.

# Microprogramada

- Em uma unidade de controle microprogramada, os sinais de controle estão armazenados em uma memória especial chamada MEMÓRIA DE CONTROLE.
- Vários sinais de controle são buscados a cada acesso à memória de controle. Estes sinais estão agrupados em longas palavras, chamadas MICROINSTRUÇÕES.
- Um conjunto delas forma um MICROPROGRAMA

# Sinais de Entrada da UC

- RST = Registrador de Estado
- RI = Registrador de Instruções
- PC = Contador de Programa



## PC – Contador de Programa

- Apontador de Instruções
- Mantém atualizado o endereço de memória da próxima instrução que deve ser executada
- Para acessar a próxima instrução de um programa, basta CONTAR mais um
- Possui comprimento em bits suficiente para endereçar todas as instruções de um programa

## RI – Registrador de Instruções

- Armazena a instrução que está sendo executada
- Em função de seu valor, a UC determina quais sinais de controle devem ser gerados para executar as operações determinadas pela instrução
- Seu comprimento em bits depende do tamanho e da codificação das instruções do computador

## RST – Registrado de Estado

- Armazena códigos de condição gerados pela ULA (ou ainda sinais de interrupção gerados por dispositivos de E/S)
- Em função dele, a UC toma decisões sobre a geração ou não de certos sinais de controle
- Seu comprimento em bits é em função do número de códigos de condição implementados na máquina

## Conjunto de instruções e modo de endereçamento

## Instrução

- Conjunto de bits devidamente codificados que indica ao computador que sequência de micro-operações ele deve realizar
- Instruções são classificadas por semelhança, propósito e formato:
  - Transferência de dados;
  - Instruções aritméticas e lógicas;
  - Instruções de teste e desvio.

## Conjunto de Instruções

- São todas as instruções que um computador reconhece (INSTRUCTION SET)
- Um PROGRAMA é composto com um conjunto de instruções

## Formato de Instrução

- OPERAÇÃO e OPERANDO
- Operação = instrução
- Operandos = maneira de realizar a instrução
- Exemplo: MOV AH,02
- MOV AH = instrução
- 02 = operando

## Programa

- Sequência de instruções de uma determinada arquitetura (Ex: x86 de 32 bits)
- Programas em alto nível podem ter portabilidade alta, ou seja, a preocupação com as instruções para uma determina arquitetura “desaparecer” (pelo menos para o programador!)

## Memória e Endereços

- Armazena:
- Instruções (Ex: printf “ou” MOV AH,02)
- Dados (Ex: int x “ou” x db 0x00)
- A memória é acessada por endereços
- As estruturas atuais permitem “segmentação” ou “paginação” da memória, sendo que o sistema operacional gerencia isso

## Modos de Endereçamento

- Muitas instruções realizam operações sobre OPERANDOS
- Endereço do operando = REGISTRADORES e ENDEREÇOS DE MEMÓRIA (variáveis);
- Endereço de programa = IF, ELSE, GOTO

## Busca – Decodificação - Execução

- Vamos agora verificar como funciona o procedimento de completo de “execução” de um instrução
- Faremos a abstração de diversos itens da eletrônica que não são do escopo desse curso

## CICLO DE BUSCA – DEC. – EXEC.

- Busca = ler uma instrução da memória;
- Decodificação = qual instrução deve ser executada;
- Execução = para cada tipo de instrução, conforme necessário.

## BUSCA

- Copiar o PC para o registrador de endereços da memória (REM);
- Ler uma instrução da memória;
- Copiar o registrador de dados da memória (RDM) para o registrador de instruções (RI);
- Atualizar o apontar/contador de programa (PC).

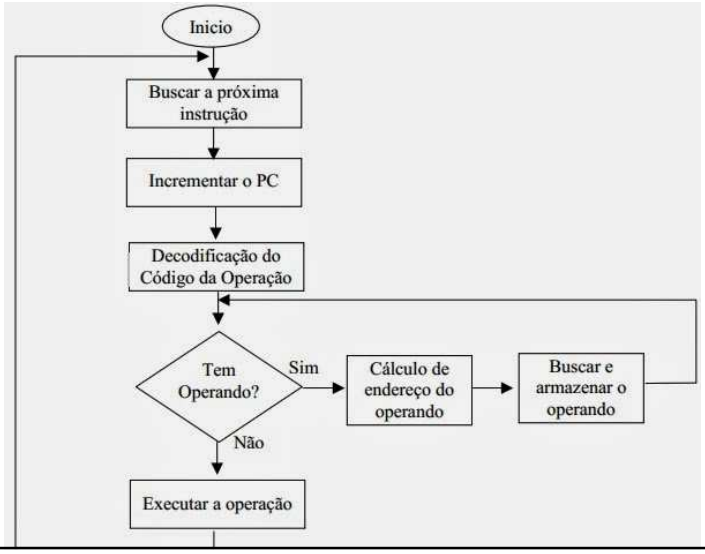
## DECODIFICAÇÃO

- A instrução é interpretada por circuitos de decodificação
- Esses circuitos geram sinais eletrônicos no processador como resultado do valor do campo de OPERAÇÃO
- Decodificam a informação correspondente à operação a ser realizada

# EXECUÇÃO

- Os sinais eletrônicos anteriores resultam na execução
- É a aplicação da função da OPERAÇÃO nos OPERADOS
- Ao final da execução o contador de instruções (PC) é atualizado para o endereço da próxima instrução
- Voltamos ao início do processo

# Fluxograma do Processo



## Programação de um Processador

- Processador compreende somente LINGUAGEM DE MÁQUINA (“imagem binária”)
- Representa a codificação do CONJUNTO DE INSTRUÇÕES do computador
- Programação complexa e de difícil depuração

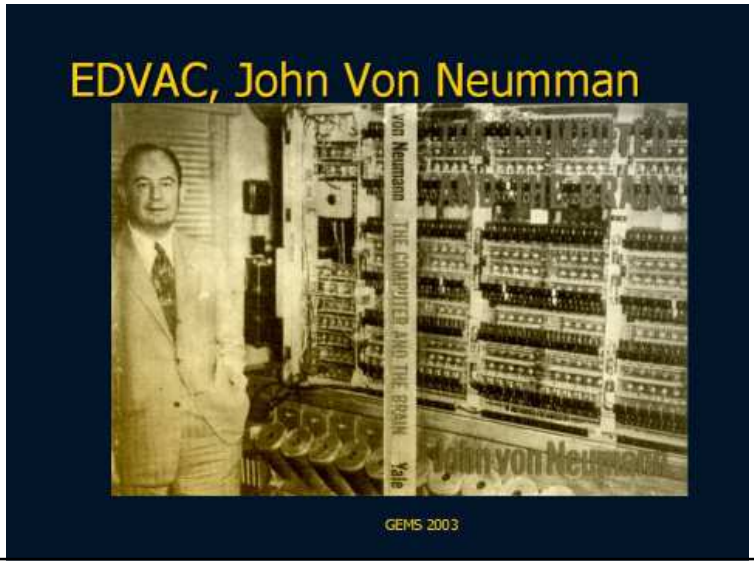
## Facilitadores de Programação

- MNEMÔNICOS (do inglês) foram associados aos códigos das instruções;
- NOME aos operandos;
- RÓTULOS (labels) às posições ocupadas pelo programa.

# Tradução para Máquina

- Linguagem “simbólica” precisa ser convertida para Linguagem de Máquina
- O MONTAR realiza esse processo, chamado de MONTAGEM
- É praticamente uma tradução UM para UM
- Um COMPILADOR faz muito mais do que isso (não abordaremos aqui)
- Os montadores possuem PSEUDOINSTRUÇÕES

# EDVAC - 1951



## EDVAC – Computador de Primeira Geração

- As instruções estavam na memória principal
- As palavras de memória com 44 bits:
  - 4 campos de endereços de 10 bits
  - 1 campo com código de instrução de 4 bits

  (16 instruções possíveis, mas 12 “criadas”)

## Instruções no EDVAC - Aritmética

- Aritmética:
- A1 A2 A3 A4 OP
- A1 e A2 = endereços de origem
- A3 = endereço de destino
- A4 = endereço da próxima instrução
- OP = Operação

## Instruções no EDVAC - Condicional

- Condicional:
- A1 A2 A3 A4 C
- Se o valor em A1 não é menor do que o valor em A2 então execute a instrução que está na posição A3
- Caso contrário, execute a instrução que está na posição A4
- C = Indica uma operação de condição

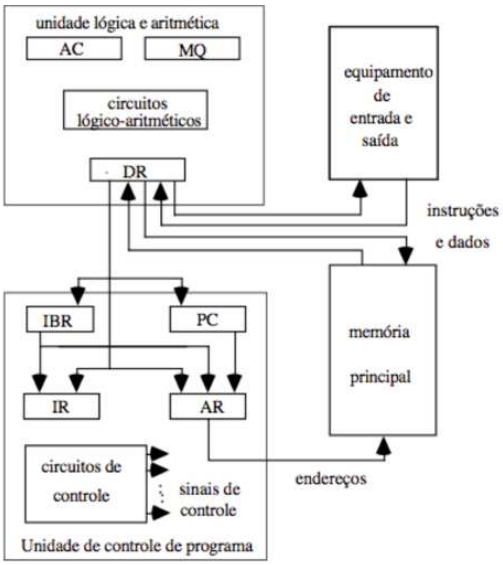
## Instruções no EDVAC – E/S

- E/S:
- A1 m,n A3 A4 W
- Se m = 1, transfira para o condutor ‘n’ na sequência de palavras armazenadas na memória principal nas posições A1, A1+1, A1+2, ...., A3.
- Se m = 2, transfira do condutor ‘n’ a sequência de palavras para as posições A1, A1+1, A1+2,..., A3 na memória principal.

# Gargalos de Von Neumann

- As instruções possuem inúmeras interações com a memória
- Algumas possibilidades para resolver:
  - Resultado da operação armazenado em um dos operandos
  - Omitir o endereço do resultado quando temos um local fixo para o resultado
  - Se o endereço da próxima instrução for convencionado, não precisa ser especificado

# IAS – 1946 (projeto)



## IAS – Instruções (parte 1 de 2)

- Formato: OP A
- AC <- M(100) = Transfere o conteúdo da memória, do endereço 100, para o acumulador
- AC <- AC + M(101) = Soma o conteúdo da posição de memória 101 ao conteúdo do acumulador e coloca o resultado no acumulador

## IAS – Instruções (parte 2 de 2)

- M(102) <- Armazena o conteúdo do acumulador no endereço 102 da memória