IAS

Arquitetura e Organização de Computadores

Prof. Lucas de Oliveira Teixeira

UEM

Introdução

Introdução

- Foi um dos primeiros computadores a implementar o conceitos de programa armazenado.
- Construído pelo Instituto de Estudos Avançados de Princeton.
- Começou a ser desenvolvido em 1945 e terminou em 1951.
- O artigo descrevendo seu projeto foi editado por John von Neumann em 1952.

Introdução

- Foi o primeiro projeto a misturar programas e dados numa única memória.
- Seguia fielmente a arquitetura de Von Neumann.
- Foi uma prova de conceitos para sua proposta.

Memória

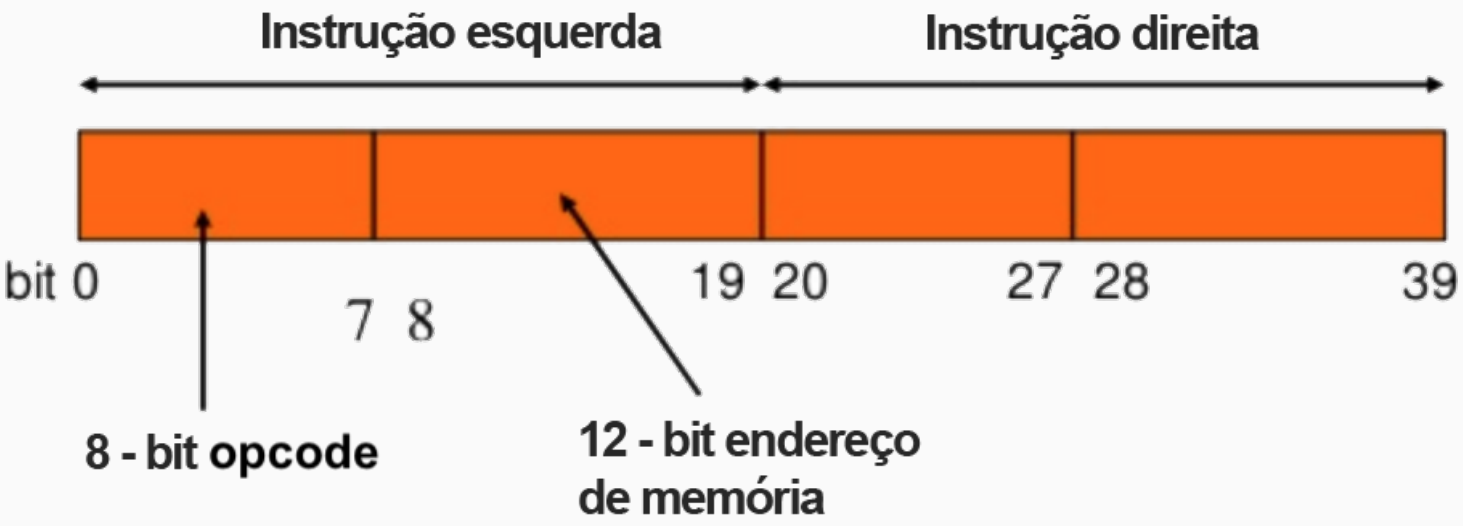
Memória

Memória:

- Palavra de 40 bits:
  - Armazenava duas instruções de 20 bits em cada palavra (8 bits para o opcode e 12 bits de operando).
  - Armazenava um dado de 40 bits em uma palavra.
- A memória tinha capacidade para 1024 palavras.
- Dois registradores de uso geral:
  - Acumulador (AC).
  - Multiplicador/Quociente (MQ).

Memória

Palavra de 40 bits:



Instruções

Instruções

Tipos de instruções:

- Transferência de dados: Instruções para mover dados entre a memória e os registradores.
- Salto: Instruções para desviar o fluxo da execução das instruções.
- Aritmética: Instruções para realização de operações aritméticas.
- Modificação de endereço: Instruções para alterar o campo endereço de outras instruções.

Instruções

Instruções de transferência de dados:

OPCODE	Instrução	Significado
00001010	LOAD MQ	AC = MQ
00001001	LOAD MQ, M(X)	MQ = mem(X)
00100001	STOR M(X)	mem(X) = AC
00000001	LOAD M(X)	AC = mem(X)
00000010	LOAD -M(X)	AC = -mem(X)
00000011	LOAD |M(X)|	AC = |mem(X)|
00000100	LOAD -|M(X)|	AC = -|mem(X)|

Instruções

Instruções de salto incondicional:

OPCODE	Instrução	Significado
00001101	JUMP M(X, 0:19)	Salta para a primeira instrução de mem(X)
00001110	JUMP M(X, 20:39)	Salta para a segunda instrução de mem(X)

Instruções

Instruções de salto condicional:

OPCODE	Instrução	Significado
00001111	JUMP +M(X, 0:19)	Se AC >= 0, salta para a primeira instrução de mem(X)
00010000	JUMP +M(X, 20:39)	Se AC >= 0, salta para a segunda instrução de mem(X)

Instruções

Instruções artiméticas:

OPCODE	Instrução	Significado
00000101	ADD M(X)	AC = AC + mem(X)
00000111	ADD |M(X)|	AC = AC + |mem(X)|
00000110	SUB M(X)	AC = AC - mem(X)
00001000	SUB |M(X)|	AC = AC - |mem(X)|
00001011	MUL M(X)	MQ = MQ * mem(X)
00001100	DIV M(X)	MQ = MQ / mem(X) e o resto fica em AC
00010100	LSH	Desloca os bits para esquerda (multiplica por 2)
00010101	RSH	Desloca os bits para direita (divide por 2)

Instruções

Instruções de alteração de endereço:

OPCODE	Instrução	Significado
00010010	STOR M(X, 8:19)	Move os 12 bits à direita de AC para o campo endereço de endereço da instrução à esquerda da palavra X na memória
00010011	STOR M(X, 28:39)	Move os 12 bits à direita de AC para o campo endereço de endereço da instrução à direita da palavra X na memória

Registradores

Registradores

Registrador	Tamanho	Função
Contador de programa (PC)	12 bits	Armazena o endereço da próxima instrução
Acumulador (AC)	40 bits	Armazenamento temporário de dados
Multiplier quotient (MQ)	40 bits	Armazenamento temporário de dados
Buffer de dados/instrução (MBR)	40 bits	Dados de leitura/escrita na memória
Buffer de instrução (IBR)	20 bits	Armazena instrução direita (bits 20-39)
Registrador de instrução (IR)	8 bits	Armazena opcode de uma instrução
Endereço de memória (MAR)	12 bits	Armazena endereço de memória de uma instrução

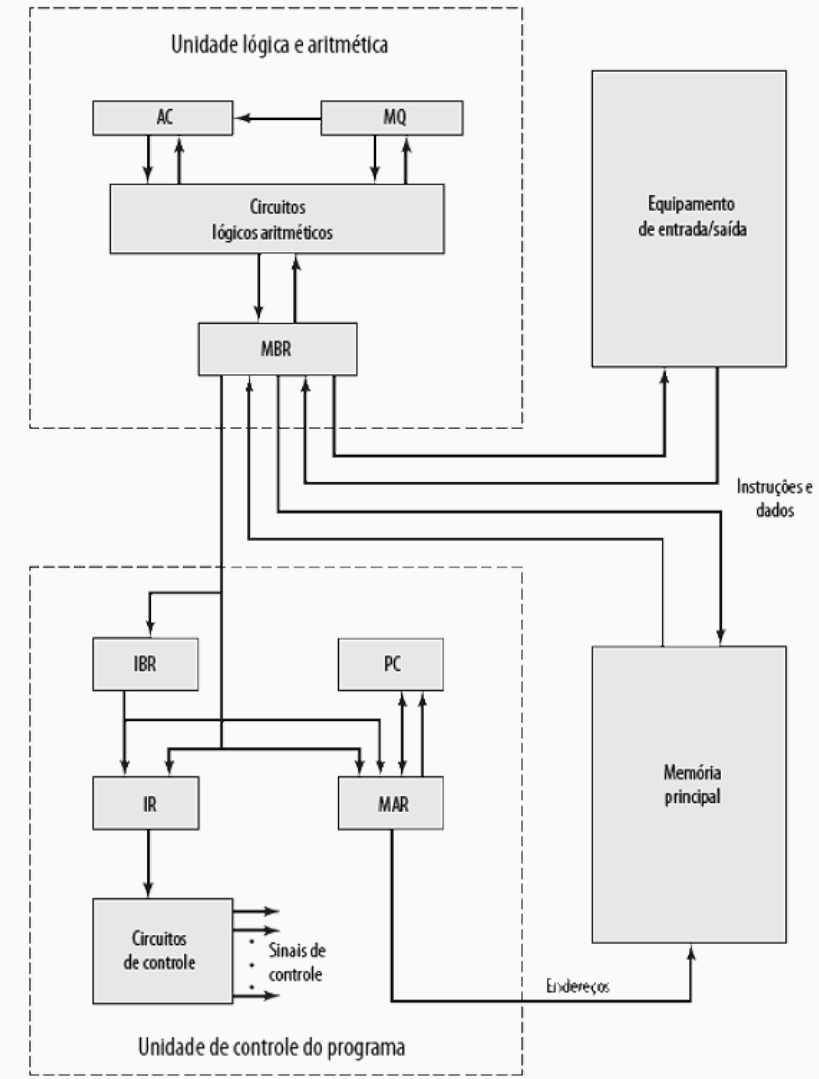
Execução

Execução

- Inicialmente, o computador executa a instrução à esquerda da primeira palavra da memória (endereço 0).
- Depois a instrução à direita da primeira palavra da memória.
- Em seguida, prossegue executando a instrução à esquerda da segunda palavra da memória (endereço 1), e assim por diante.
- Obs.: É válido lembrar que algumas instruções podem desviar o fluxo de execução. Neste caso a próxima instrução a ser executada não será a instrução subsequente na memória.

Execução

Estrutura IAS:



Execução

A execução de uma instrução é realizada em dois ciclos:

- Ciclo de busca: Consiste em buscar a instrução da memória (ou do registrador IBR) e armazenar no IR.
- Ciclo de execução: Consiste em interpretar a instrução armazenada no registrador IR e realizar as operações necessárias para execução da mesma.

Execução

Ciclo de busca:

- A UC move o endereço em PC para MAR;
- A UC envia um sinal de controle para a memória fazer uma operação de leitura;
- A memória lê a palavra contida no endereço que está em MAR e a transfere para o registrador MBR;
- A UC copia a segunda metade da palavra (bits 20 a 39) do registrador MBR e salva no registrador IBR. Estes bits correspondem à instrução à direita.

Execução

Ciclo de busca:

- A UC copia os 8 bits à esquerda do registrador MBR para o registrador IR. Estes bits correspondem ao campo de operação (opcode) da instrução à esquerda.
- A UC copia os 12 bits subsequentes ao campo de operação (bits 8 a 19) e os transfere para o registrador MAR. Estes bits correspondem ao campo endereço na instrução. Devem estar no registrador MAR caso a instrução precise acessar a memória durante a execução.
- A UC incrementa o valor de PC, indicando que o próximo par de instruções a ser lido da memória deve ser lido do endereço PC + 1.

Execução

Ciclo de execução:

- Cada instrução executa passos distintos durante sua execução. Obviamente, senão elas teriam a mesma funcionalidade.
- Cabe aos circuitos de controle:
  - Identificar a instrução armazenada no registrador IR; e
  - Coordenar a execução dos passos apropriados para a execução dessa instrução.

Execução

Ciclo de execução do ADD:

- A UC interpreta os bits armazenados em IR (00000101 no caso da instrução ADD M(X)) e identifica a instrução como sendo uma soma.
- A UC sabe que a instrução requer a busca de operandos da memória, desta forma:
  - A UC envia um sinal para a memória realizar uma operação de leitura. O endereço do operando já foi transferido para o registrador MAR durante o ciclo de busca.
  - A memória lê a palavra e transfere o conteúdo para o registrador MBR;

Execução

Ciclo de execução do ADD:

- A UC sabe que a instrução ADD envolve a realização de uma operação de soma na ULA, então:
  - A UC envia sinais para a ULA solicitando a realização da soma dos valores armazenados em AC e MBR (AC = AC + MBR). Neste ponto os operandos já se encontram em AC e MBR.
  - A ULA realiza a operação de soma.
  - A ULA grava o resultado da soma no registrador AC.

Execução

Por exemplo, faça um programa usando o Assembly IAS para somar dois valores localizados na memória nos endereços 100 e 101 e guarde o resultado na posição 110 da memória.

```
1  load M(100)
2  add M(101)
3  stor  M(110)
```

Execução

Por exemplo, faça um programa usando o Assembly IAS para realizar a execução da seguinte expressão: (235 * 3) + (899 * 23). Considere que os quatro valores inteiros estão armazenados nos endereços 100 até 103 respectivamente na ordem em que aparecem.

```
1  load MQ, M(100)
2  mul M(101)
3  load MQ
4  stor  M(104)
5  load MQ(102)
6  mul M(103)
7  load MQ
8  add M(104)
9  stor  M(104)
```

Execução

Por exemplo, faça um programa usando o Assembly IAS para calcular o fatorial de um número **n**.

```
1  int n = 10, fat = 1, i = 1;
2  while (i <= n) {
3    fat = fat * i;
4    i = i + 1;
5  }
```

Considere que a variável n está na posição 100, fat na posição 101, i na posição 102 e a constante 1 na posição 103.

Execução

```
1  start :
2  load MQ, M(101)
3  mul M(102)
4  load MQ
5  stor  M(101)
6  load M(102)
7  add M(103)
8  stor  M(102)
9  load M(100)
10 sub M(102)
11 jump +M(start)
```