Instituto Federal de Santa Catarina
Área de Telecomunicações

MIC29004 – Microprocessadores

Prof. Roberto de Matos

roberto.matos@ifsc.edu.br

São José, agosto de 2013.

Definições

Máquina de níveis

Princípios Básico

Cada computador tem um conjunto de operações e convenções únicas para determinar as posições de dados com os quais a operação será realizada:

OPERAÇÃO	OPERANDOS

Isso é denominada de instruções:

OPERAÇÃO → especifica a função que será desempenhada;

OPERANDOS → fornece a maneira de calcular a posição atual dos dados com os quais a OPERAÇÃO será realizada.

- *Um **programa** é constituído de uma sequencia pré-determinada de instruções, que deve ser seguida para que seja atingido uma tarefa computacional.*

Princípios Básicos

- *A **memória** de um sistema computacional tem a função de armazenar dados e instruções.*
  - Organizada em posições;
  - Podem ser visualizadas como elementos de uma matriz;
  - Cada elemento tem um **endereço**.
- Então uma memória que tenha *x* posições:
  - Cada posição pode ser referenciada diretamente de acordo com a sua colocação na seqüência;
  - Se uma memória tem 4096 posições existem posições de 0, ..., 4095;
  - Instruções são executadas em uma seqüência determinada pela sua posição de memória

Princípios Básicos

- *O **endereço** representa uma posição particular na memória e pode ser formado de várias maneiras;*
- A representação trivial está na parte chamada **campo de endereço**;
- A Unidade Lógica e Aritmética (ULA) é responsável por realizar ações indicadas nas instruções, executando operações numéricas (aritméticas) e não numéricas (lógica);
  - Preparação de informações de desvios do programa;
- O controle do programa e a ULA formam a **unidade central de processamento**, ou simplesmente processador.

Computadores não pensam!!

- O programador do computador fornece um programa composto por instruções e dados que especificam cada detalhe do que executar, do que fazer com os dados, e quando fazê-los.
- O computador é, simplesmente, uma máquina de alta velocidade que pode manipular dados, resolver problemas e tomar decisões, tudo sob o controle do programa.
- Se o programador cometer um erro no programa ou fornecer dados errados, o computador invariavelmente fornecerá saídas erradas, como na máxima "lixo entra, lixo sai"

- Talvez uma melhor questão a ser feita neste ponto seja: Como um computador consegue executar um conjunto de instruções?
- Tipicamente, respondemos a esta questão mostrando o diagrama da arquitetura de um computador (arranjo de seus vários elementos) e, então, repassando o processo, passo a passo, que o computador segue na execução do programa.
- Ao invés disso, primeiro iremos observar uma analogia um pouco distante, que contém muitos dos conceitos envolvidos na operação de um computador.

Agente 89 do GOE

- O Agente da GOE está tentando descobrir quantos dias faltam para um certo líder ser assassinado. O seu contato lhe diz que esta informação está localizada em uma série de caixas postais. Para assegurar que ninguém mais tenha acesso à informação, ela está espalhada em 10 caixas. O seu contato lhe dá as 10 chaves, acompanhadas das seguintes instruções:

- 1. A informação em cada caixa está escrita em código
- 2. Abra a caixa 1, primeiro, e execute a instrução lá localizada.
- 3. Continue pelo restante das caixas, em seqüência, salvo instrução em contrário.
- 4. Uma das caixas está preparada para ser explodida ao ser aberta.

- O Agente pega as 10 chaves e vai para a agência do correio, com o livro de códigos em mãos.

Tabela

(1)	*(2)*
(3)	*(4)*
(5)	*(6)*
(7)	*(8)*
(9)	*(10)*

Tabela

(1) Some o número armazenado na caixa (9) ao seu número de código secreto	*(2)*
(3)	*(4)*
(5)	*(6)*
(7)	*(8)*
(9)	*(10)*

Tabela

(1) Some o número armazenado na caixa (9) ao seu número de código secreto	*(2)*
(3)	*(4)*
(5)	*(6)*
(7)	*(8)*
(9)	*(10)*

Tabela

(1) Some o número armazenado na caixa (9) ao seu número de código secreto	***(2)*** Divida o resultado pelo número armazenado na caixa (10)
(3)	***(4)***
(5)	***(6)***
(7)	***(8)***
(9) 11	***(10)***

Tabela

(1) Some o número armazenado na caixa (9) ao seu número de código secreto	***(2)*** Divida o resultado pelo número armazenado na caixa (10)
(3)	***(4)***
(5)	***(6)***
(7)	***(8)***
(9) 11	***(10)*** 2

Tabela

(1) Some o número armazenado na caixa (9) ao seu número de código secreto	***(2)*** Divida o resultado pelo número armazenado na caixa (10)
(3) Subtraia o número armazenado na caixa (8)	***(4)***
(5)	***(6)***
(7)	***(8)***
(9) 11	***(10)*** 2

Tabela

(1) Some o número armazenado na caixa (9) ao seu número de código secreto	***(2)*** Divida o resultado pelo número armazenado na caixa (10)
(3) Subtraia o número armazenado na caixa (8)	***(4)***
(5)	***(6)***
(7)	***(8)*** 20
(9) 11	***(10)*** 2

Tabela

(1) Some o número armazenado na caixa (9) ao seu número de código secreto	***(2)*** Divida o resultado pelo número armazenado na caixa (10)
(3) Subtraia o número armazenado na caixa (8)	***(4)*** Se o resultado anterior não for igual a 30, vá para a caixa (7); caso contrário, vá para a caixa seguinte.
(5)	***(6)***
(7)	***(8)*** 20
(9) 11	***(10)*** 2

Tabela

(1) Some o número armazenado na caixa (9) ao seu número de código secreto	***(2)*** Divida o resultado pelo número armazenado na caixa (10)
(3) Subtraia o número armazenado na caixa (8)	***(4)*** Se o resultado anterior não for igual a 30, vá para a caixa (7); caso contrário, vá para a caixa seguinte.
(5) Subtraia 13 do resultado anterior	***(6)***
(7)	***(8)*** 20
(9) 11	***(10)*** 2

Tabela

(1) Some o número armazenado na caixa (9) ao seu número de código secreto	*(2)* Divida o resultado pelo número armazenado na caixa (10)
(3) Subtraia o número armazenado na caixa (8)	*(4)* Se o resultado anterior não for igual a 30, vá para a caixa (7); caso contrário, vá para a caixa seguinte.
(5) Subtraia 13 do resultado anterior	*(6)* PARE. Você já tem a resposta.
(7)	*(8)* 20
(9) 11	*(10)* 2

Tabela

(1) Some o número armazenado na caixa (9) ao seu número de código secreto	*(2)* Divida o resultado pelo número armazenado na caixa (10)
(3) Subtraia o número armazenado na caixa (8)	*(4)* Se o resultado anterior não for igual a 30, vá para a caixa (7); caso contrário, vá para a caixa seguinte.
(5) Subtraia 13 do resultado anterior	*(6)* PARE. Você já tem a resposta.
(7) BOMBA!!!	*(8)* 20
(9) 11	*(10)* 2

Conclusões (computacionais)

- Três classes diferentes de instruções estão presentes nas caixas de 1 a 6. As caixas 1, 2, 3 e 5 são instruções que "chamam" *operações aritméticas*.
- A caixa 4 contém uma instrução de *decisão*, chamada de desvio condicional.
- Esta instrução diz ao agente (ou computador) para decidir se pula para o endereço 7 ou continua para o endereço 5, dependendo do resultado da operação aritmética anterior.
- A caixa 6 contém uma simples instrução de controle que não precisa de dados, nem faz referência a outro endereço (número de caixa).
- Essa instrução, *pare*, diz ao agente que o problema está terminado (o programa está completado) e que não deve continuar adiante.

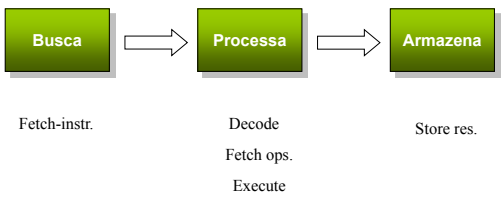
Busca – Decodificação - Execução

- Um elemento no processador denominado **contador de instruções (PC)**, contém a posição da próxima instrução a ser executada;
- Quando uma sequencia de execução de instrução tem início, a instrução cujo endereço está no PC é trazida da memória para uma área de armazenamento chamada **registrador de instrução (IR)**.
  - Chamado de **busca**
- A instrução é interpretada por circuitos de decodificação que fazem com que os sinais eletrônicos sejam gerados no processador
  - Chamado de **decodificação**

Busca – Decodificação - Execução

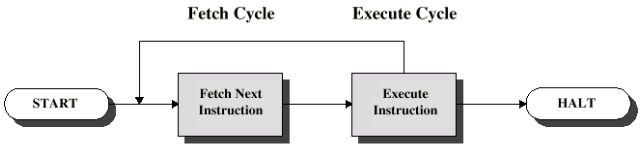
- Esses sinais resultam na **execução**. Execução é a aplicação da função do operador nos operandos;
- Quando uma execução de uma instrução é terminada, o contador de instruções é atualizada para o endereço de memória para próxima instrução;
- Esta instrução é então trazida da memória para o IR e executada, repetindo assim o ciclo de ***busca-decodificação-execução***;

Esquema Básico



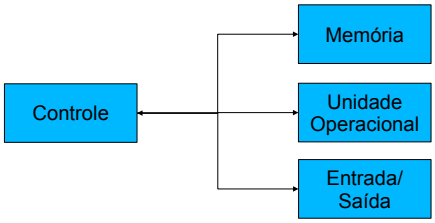
Esquema Básico

- Dois passos:
  - Buscar
  - Executar



Elementos funcionais básicos

- Computador:
  - Composto por blocos:
    - Memória
    - Unidades operacionais
    - Unidades de controle
    - Dispositivos de entrada e saída



Elementos funcionais básicos

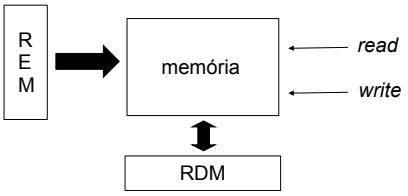
- **Unidade Operacional** e **Unidade de Controle** tem funções específicas;
- Reunidas recebe o nome de UCP – Unidade Central de Processamento;
- **Registradores** são elementos digitais com capacidade de armazenar dados.
  - Tem associados sinais de cargas que determinam quando serão armazenados novos conteúdos neles;
  - Ao ser acionado o sinal de carga, o registrador copia para si o dado que está em suas linhas de entrada.

Elementos funcionais básicos

- **Contadores**, **multiplexadores**, **seletores**, **codificadores**, **somadores** e **portas lógicas** são elementos com capacidade de operar sobre dados:
  - Alterando-os ou fornecendo um novo dado como resultado
- **Elementos digitais**
  - Necessitam ser ativados ou habilitados para realizar uma determinada operação;
  - Sinais responsáveis pela ativação ou habilitação de componentes digitais são conhecidos como **sinais de controle**.
- **Barramentos *(bus)***:
  - Transferem dados entre os elementos do computador;
  - Só podem receber dados de uma fonte de cada vez;
  - A largura em bits diz respeito ao tamanho dos elementos que trafegam nele.

Elementos funcionais básico - memória

- Memória é formada por elementos armazenadores de informações;
- É dividida em **palavras**;
- Cada palavra é identificada unicamente por um **endereço**;
- Conteúdo armazenado nas palavras da memória tanto pode representar dados como instruções;



Elementos funcionais básico - memória

REM: registrador de endereços da memória

Contém o endereço do dado a ser lido ou escrito na memória

RDM: registrador de dados da memória

Contém o dado a ser escrito na memória (*write*) ou lido da memória (*read*)

Sinais de controle:

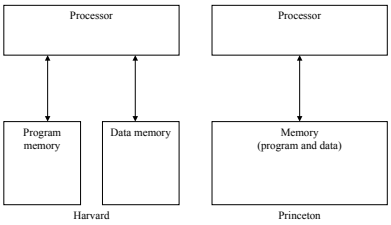
Read: leitura da memória – o conteúdo da posição de memória endereçada por REM é copiado em RDM;

Write: escrita na memória – a posição de memória endereçada por REM recebe o conteúdo de RDM;

Arquiteturas de Memória

Princeton
- Menos fios

Harvard
- Acesso simultâneo à dado e instruções



Elementos funcionais básico – Unidade Operacional

- Unidade Operacional:
  - Também conhecida como bloco operacional (*datapath*);
  - Executa as transformações sobre dados, especificadas pelas instruções do computador;
  - Componentes:
    - ULA, registradores de uso geral e específico e barramentos
  - O número; tamanho e uso dos registradores e a quantidade; e tipo de operações que a ULA realiza são alguns dos fatores que influenciam no poder de processamento de um computador.
    - Vamos ver a **ULA** e o **Acumulador**

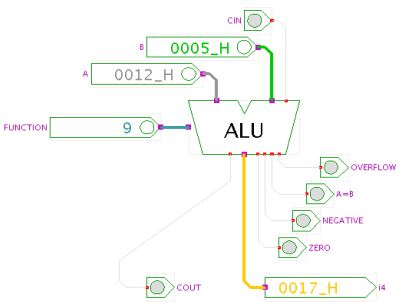
Elementos funcionais básico – Unidade Operacional

- Unidade Lógica e Aritmética (ULA)
  - Realiza operações lógicas e aritméticas;
  - Exemplo:
    - Soma de dois operandos;
    - Negação de um operando;
    - Inversão de um operando;
    - Lógica de operando;
    - Rotação de um operando para a direita ou esquerda.
  - As operações da ULA geralmente são bem simples;
  - Funções complexas são realizadas pela ativação sequencial das várias operações básicas;

Elementos funcionais básico – Unidade Operacional

- A ULA fornece o resultado das operações e também algumas indicações sobre a operação realizada;
- Essas indicações são chamadas de códigos de condição;
- Exemplos:
  - *Overflow*
  - *Sinal*
  - *Carry*
  - *Zero*

Modelo estrutural ULA



Elementos funcionais básico – Unidade Operacional

- **Acumulador**
  - É um registrador e tem por função armazenar um operando e/ou um resultado fornecido pela ULA;
  - Computadores simples só são encontrados um acumulador;
  - É ativado de acordo com o sinal de carga (*load*);
    - Cada novo sinal faz perder o valor antigo

Elementos funcionais básico – Unidade de Controle

- Serve para fornecer **sinais de controle**:
  - Gerenciar o fluxo interno de dados e o instante preciso em que ocorrem as transferências entre uma unidade e outra
- Cada unidade de controle comanda uma micro operação
  - Responsável pela realização de uma carga em um registrador;
  - Seleção de dados para entrada;
  - Ativação de memória;
  - Seleção de uma operação da ULA.

Elementos funcionais básico – Unidade de Controle

- São máquinas de Estado Finita (FSM)
  - Lógica sequencial: os sinais de saída são função dos sinais de entrada e do estado anterior.
  - Lógica combinacional: os sinais de saída são função exclusiva dos sinais de entrada.
- Existem várias formas de implementar a lógica sequencial. Porém duas são usuais:
  - Organização convencional
  - Organização microprogramada

Elementos funcionais básico – Registradores especiais

- Existem no computador alguns registradores com funções especiais:
  - Depende da arquitetura e organização de cada máquina;
  - A posição também influencia.
- Tipos:
  - **Apontador de instruções (PC)**
  - **Registrador de instruções (IR)**
  - **Registrador de estado (RST)**

Elementos funcionais básico – Registradores especiais

Apontador de Instruções (PC)
Tem como função manter atualizado o endereço de memória da próxima instrução;

Registrador de instrução (IR)
Armazena a instrução que está sendo executada;
De acordo com o conteúdo, a UC determina quais sinais deve mser gerados;

Registrador de estado (RST)
Armazena códigos de condição gerados pela unidade lógica e aritmética;

Conjunto de instruções e modos de endereçamento

- É um conjunto de bits devidamente codificados que indica ao computador que sequencia de micro operações este deve seguir:
- São classificadas por semelhanças de propósito e formato:
  - Instruções de transferência de dados;
  - Instruções aritméticas e lógicas;
  - Instruções de testes e desvios.
- Logo **conjunto de instruções** são todas as instruções que um determinado computador reconhece;
- A sequencia dessas instruções forma um programa.

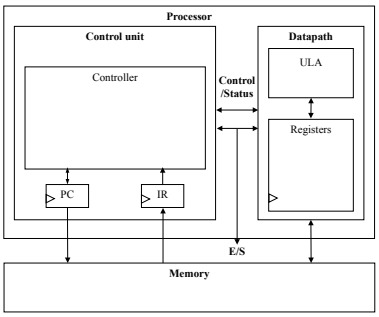
Instruções do Microprocessador

- Esta coleção de palavras é conhecida como a **linguagem *assembly*** do processador.
- Um ***assembler*** (montador) pode traduzir as palavras para o seu padrão binário e a informação de saída do *assembler* é alocada na memória para ser executada pelo microprocessador.
- Para facilitar as tarefas de programação e depuração:
  - Mnemônicos → associados aos códigos das instruções
  - Nomes → operandos
  - Rótulos → posições ocupadas pelo programa

Arquitetura Básica

Unidade de Controle e de Processamento;

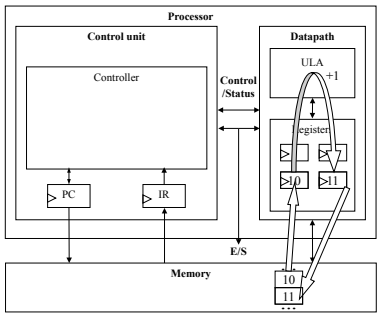
Diferenças
- Unidade de Processamento é genérica
- Unidade de controle não armazena algoritmo (memória)



Operações de Processamento

Load
Cópia de memória em registrador

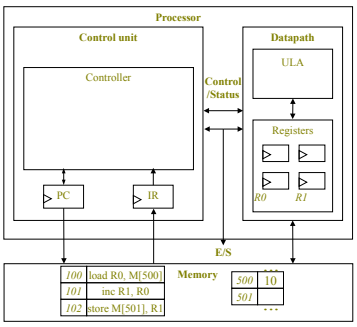
- Operação na ULA
  - Valores em Registradores são processados pela ULA e armazenados em registrador
- Armazena
  - Cópia de registrador em memória



Unidade de Controle

Unidade de Controle: configura operações do datapath
- Seqüência de operações (instruções) desejadas armazenadas na memória (programa)

Ciclo de Instrução – várias sub-operações (cada uma em um ciclo de relógio)
- **Busca**: armazena instrução em IR, atualiza PC
- **Decodificação**: determina o que a instrução significa
- **Busca de Operandos**: cópia de dados da memória para registradores na unid. Processamento
- **Execução**: Processa dados na ULA
- **Armazena resultados**: escreve resultados de registrador na memória

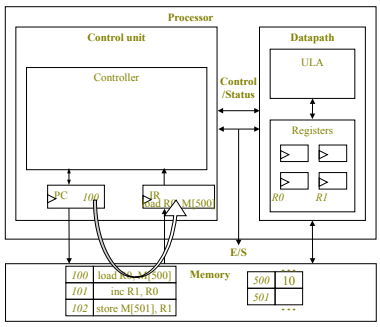


Sub-operações da Unidade de Controle

Busca
Cópia da instrução em IR

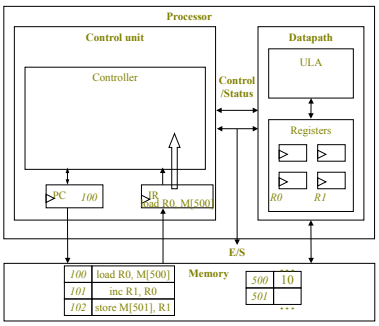
PC: Contador de programa aponta para próxima instrução

IR: armazena instrução que foi buscada



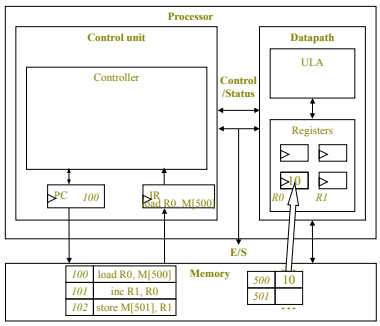
Sub-operações da Unidade de Controle

Decodificação
Determina significado da instrução



Sub-operações da Unidade de Controle

Busca de Operandos
Cópia de dados da memória para registradores no *datapath*

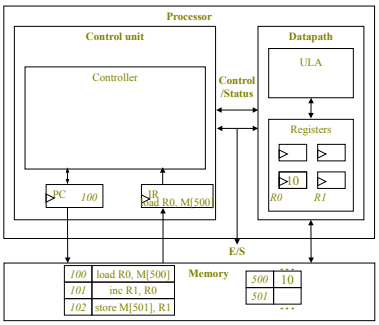


Sub-operações da Unidade de Controle

Execução

Processa dados na ULA

(Para esta instrução em particular nada acontece durante esta sub-operação)

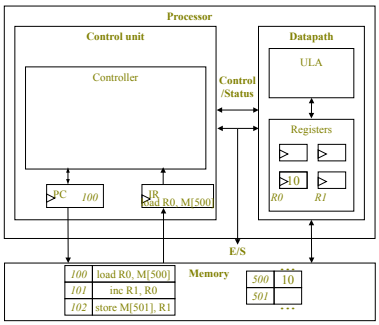


Sub-operações da Unidade de Controle

Armazena resultados

Escreve dado de registrador em memória

(Para esta instrução em particular nada acontece durante esta sub-operação)

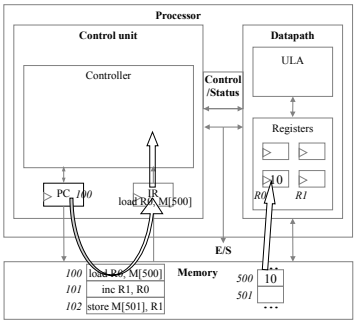


Ciclos de uma Instrução

PC=100

Fetch Decode Fetch ops Exec. Store results

clk



Ciclos de uma Instrução

PC=100

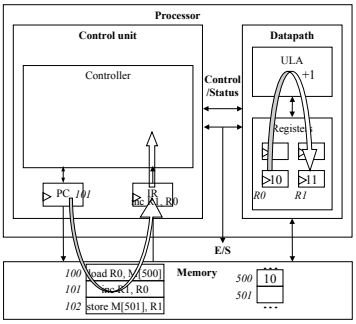
Fetch Decode Fetch ops Exec. Store results

clk

PC=101

Fetch Decode Fetch ops Exec. Store results

clk



Ciclos de uma Instrução

PC=100

Fetch Decode Fetch ops Exec. Store results

clk

PC=101

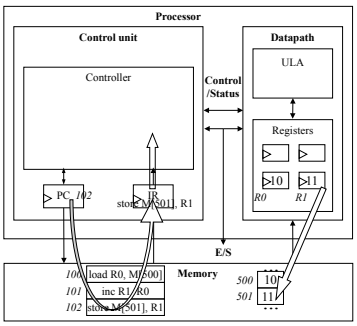
Fetch Decode Fetch ops Exec. Store results

clk

PC=102

Fetch Decode Fetch ops Exec. Store results

clk

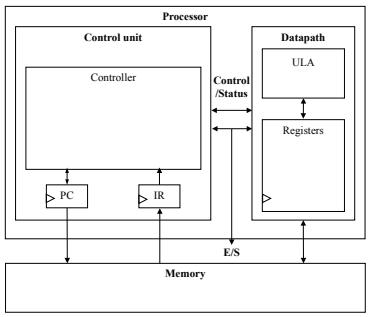


Considerações da Arquitetura

Processador de N-bits

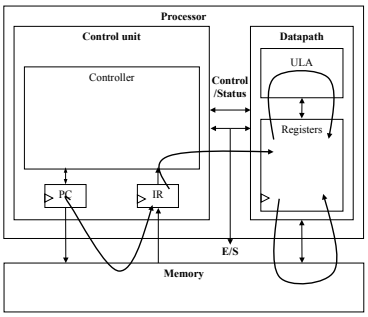
- ULA, registradores, barramento, interface de memória N-bits
- Comum em aplic. emb: 8-bit, 16-bit, 32-bit
- Comum em Desktop/servidores : 32 ou 64 bit

Tamanho do PC determina espaço de endereçamento



Considerações da Arquitetura

Freqüência do *Clock*
- Deve ser maior que o maior retardo de carregamento de registrador
- Acesso à memória possui o maior retardo



Modelo de Von Neumann

Computador IAS

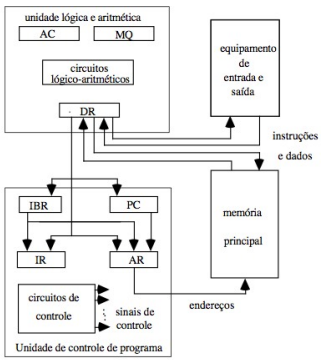
Histórico

- Projeto começou em 1946 pela equipe de von Neumann;
- Elaborado no IEAP (Instituto de estudos avançados de princeton;
- Tinha como memória principal:
  - Tubo de raios catódicos de acesso aleatório;
  - Permitia o acesso a uma palavra inteira em uma única operação;
  - Cada instrução tinha um único endereço de memória e tinha o formato:

O computador IAS

- Os blocos básicos do IAS:
  - Uma unidade de processamento central, para execução de operações lógicas e aritméticas;
  - Uma unidade de controle de programa para determinar o sequenciamento das instruções a serem executadas e gerar os sinais d controle para as outras unidades;
  - Uma unidade de memória principal com capacidade de 4096 palavras;
  - Uma unidade de E/S.

O computador IAS - Ligações



O computador IAS - Ligações

Na UCP existe um novo elemento de armazenamento de dados, que é o acumulador (AC). Esse elemento atua como memória rápida que guarda de forma imediata o resultado das operações realizadas na unidade de cálculos.

Exemplo de Programa do IAS

Instrução	Comentários
AC ← M(100)	Transfere conteúdo da memória do endereço 100 para o acumulador
AC ← AC + M(101)	Soma o conteúdo da posição de memória 101 ao conteúdo acumulador e coloca o resultado no acumulador
M(102) ← AC	Armazena o conteúdo do acumulador no endereço 102 da memória

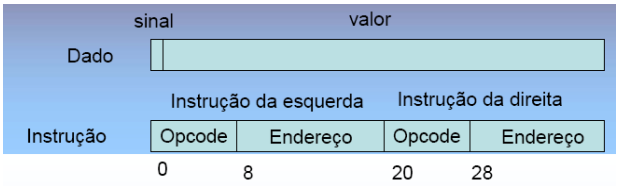
O computador IAS – Organização da Memória

- Memória composta por 2^{12} = 4096 palavras;
- Cada palavras = 40 bits;
- 40 bits eram a quantidade de informação que podiam ser transferidas, em cada momento:

  Memória → UCP (cada passo)
- Na memória:
  - Instruções de 20 em 20 bits;
  - Dados de 40 bits.

Formato dos dados e instruções

Os dados eram números binários representados em ponto fixo e codificados em complemento de dois;



Conjunto de Instruções

Tipo de Instrução	Operação	Descrição
Transferência de dados	AC ← MQ	Transfere conteúdo do reg. MQ para o acumulador AC
	MQ ← M(X)	Transfere conteúdo da posição X de memória para MQ
	M(X) ← AC	Transfere conteúdo de AC para posição de memória X
	AC ← M(X)	Transfere o conteúdo da posição de memória X para AC
	AC ← -M(X)	Transfere -M(X) para AC
	AC ← |M(X)|	Transfere o valor absoluto de M(X) para AC
	AC ← -|M(X)|	Transfere -|M(X)| para AC

Conjunto de Instruções

Tipo de Instrução	Operação	Descrição
Desvio incondicional	Vai para M(X, 0:19)	Busca a próxima instrução da metade esquerda de M(X)
	Vai para M(X, 20:39)	Busca a próxima instrução da metade direita de M(X)
Desvio condicional	Se AC >= 0 então desvia para M(X,0:19)	Se o número de AC não for negativo busca a próxima instrução na metade esquerda de M(X)
	Se AC >= 0 então desvia para M(X,20:39)	Se o número de AC não for negativo busca a próxima instrução na metade direita de M(X)
Modificação de endereço	M(X,8:19) ← AC(28:39)	Substitui o campo de endereço à esquerda de M(X) pelos 12 bits mais a direita de AC
	M(X,28:39) ← AC(8:19)	Substitui o campo de endereço à direita de M(X) pelos 12 bits mais a direita de AC
	AC ← -|M(X)|	Transfere -|M(X)| para AC

Conjunto de Instruções

Tipo de Instrução	Operação	Descrição
Aritmética	AC←AC+M(X)	Soma M(X) a AC; resultado em AC
	AC←AC+|M(X)|	Soma |M(X)| a AC; resultado em AC
	AC←AC-M(X)	Subtrai M(X) de AC; resultado em AC
	AC←AC-|M(X)|	Subtrai |M(X)| de AC; resultado em AC
	AC,MQ←MQ*M(X)	Multiplica M(X) por MQ; colocando os bits mais significativos do resultado em AC e os menos significativos em MQ
	MQ,AC←AC%M(X)	Divide AC por M(X); colocando o quociente em MQ e o resto em AC
	AC←ACx2	Multiplica AC por 2; ou desloca AC um bit a esquerda
	AC←AC/2	Divide AC por 2; ou desloca AC um bit a direita

Instruções Assembly

LOAD B mem – carrega o registrador B do endereçamento de memória
CON B con – carrega um valor constante no registrador B
SAVE B mem – armazena o registrador B no endereçamento de memória
ADD B A – soma A com B e armazena o resultado em mem
SUB B A – subtrai A de B e armazena o resultado em mem
MUL B A – multiplica A por B e armazena o resulado em mem
DIV A B – divide A por B e armazena o resultado em mem
COM A B – compara A com B e armazena o resultado no registrador teste
JUMP addr – desvia para um endereçamento
JEQ addr – desvia, se igual, para o endereçamento
JNEQ addr – desvia, se não igual, para o endereçamento
JG addr – desvia, se maior que, para o endereçamento
JGE addr – desvia, se maior que ou igual, para o endereçamento
JL addr – desvia, se menor que, para o endereçamento
JLE addr – desvia, se menor que ou igual, para o endereçamento
STOP – pára a execução

Exemplo de como funciona o *Assembly*

```
a=1;
    f=1;
    n=<entrada do usuário>
    while (a <= n)
    {
          f = f * a;
          a = a + 1;
    }
```

Se n fosse igual a cinco, f do programa seria quanto?

Linguagem *Assembly*

Um **compilador C** traduz o código em C para a linguagem *assembly*. Se considerarmos que a RAM começa no endereço 128 deste processador e a ROM (que contém o programa em linguagem *assembly*) começa no endereçamento 0.

Como funciona em *Assembly*

```
// Suponha que a está no ender. 128
// Suponha que f está no ender. 129
0 CON B 1 // a=1;
1 SAVE B 128
2 CON B 1 // f=1;
3 SAVE B 129
4 LOAD A 128 // if a > 5 desvia para 17
5 CON B 5
6 COM A B
7 JG 17
8 LOAD A 129 // f=f*a;
9 LOAD B 128
10 MUL A B
11 SAVE mem 129
12 LOAD A 128 // a=a+1;
13 CON B 1
14 ADD A B
15 SAVE mem 128
16 JUMP 4 // volta para o if
17 STOP
```

Pergunta

"Como essas instruções vão ser exibidas na ROM?"

Cada uma dessas instruções de linguagem *assembly* tem de ser representadas por um número binário. Para simplificar as coisas, vamos supor que cada instrução de linguagem *assembly* equivale a um único número.

Associação

LOAD A – 1
LOAD B – 2
CON B – 3
SAVE B – 4
SAVE C mem – 5
ADD B A – 6
SUB B A – 7
MUL A B – 8
DIV A B – 9
COM – 10
JUMP addr – 11
JEQ addr – 12
JNEQ addr – 13
JG addr – 14
JGE addr – 15
JL addr – 16
JLE addr – 17
STOP – 18

Esses números são conhecidos como **opcodes** **(códigos de operação já visto na aula passada)**. Como estaria o programa na ROM???

Programa na ROM

```
// Suponha que a está no endereçamento 128
// Suponha que f está no endereçamento 129
Addr opcode/value
0 3 // CONB 1
1 1
2 4 // SAVEB 128
3 128
4 3 // CONB 1
5 1
6 4 // SAVEB 129
7 129
8 1 // LOADA 128
9 128
10 3 // CONB 5
11 5
12 10 // COM
13 14 // JG 17
14 31
15 1 // LOADA 129
16 129
17 2 // LOADB 128
18 128
19 8 // MUL
20 5 // SAVEC 129
21 129
22 1 // LOADA 128
23 128
24 3 // CONB 1
25 1
26 6 // ADD
27 5 // SAVEC 128
28 128
29 11 // JUMP 4
30 8
31 18 // STOP
```

Conclusão sobre o programa assembly

Sete linhas de código em C = 18 linhas de linguagem *assembly* = 32 bytes na ROM.

IASSim

http://www.cs.colby.edu/djskrien/IASSim/

Dúvidas