SEL-0415 Introdução à Organização de Computadores

SEL/EESC-USP Grupo de Sistemas Digitais

Set de Instruções e Modelos de Arquiteturas

Aula 7

Prof. Dr. Marcelo Andrade da Costa Vieira

INSTRUÇÕES

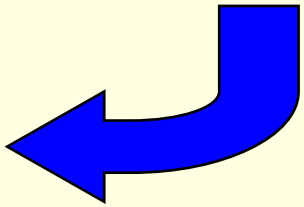
- Padrão de código binário armazenado em um dispositivo de memória para comandar o microprocessador na execução de determinada tarefa:
  - *Microcontroladores*: gravado na memória de programa (tipo ROM);
  - *PC*: gravado na BIOS (tipo ROM), que geralmente busca o Sistema Operacional gravado na memória secundária (*.exe) e carrega na memória principal (RAM) para ser executado;
- Cada μP tem seu próprio conjunto de instruções ➡ um programa em Linguagem Assembly é específico para um determinado microprocessador.

Conjunto de Instruções - Programa

- Cada µP possui suas próprias palavras binárias, significados e linguagem

- Fabricante: seleciona combinações de padrões de bits e atribui-lhes um significado utilizando circuitos lógicos específicos

SET DE INSTRUÇÕES



Set de Instruções - Programa

- Dificuldade em programar em 0 e 1: fabricantes criaram curtas palavras em inglês que representam a instrução binária na máquina (*MNEMÔNICOS*)
- Instruções Binárias ➡ Linguagem de Máquina ➡ pode ser representada na linguagem *Assembly*
- Linguagens de alto nível: Fortran, Pascal, C, Basic... (compilador → linguagem de máquina)

Set de Instruções - Programa

- Toda instrução contém um **código da operação** (OPCODE) e pode conter um ou mais **operandos**;

- Os operandos geralmente são **dados** ou **endereços**;
- Algumas instruções não possuem operandos;
- O tamanho do OPCODE e dos operandos dentro da palavra de instrução varia de acordo com a arquitetura do µP;
  - Ex.: máquina com OPCODE de 8 bits pode ter até 256 (2^8) tipos de instruções diferentes

Programa em Linguagem Assembly

- cada instrução do **μP** tem um **código binário** **(opcode)** a ela associado, que especifica a função da instrução e seus operandos.
- os fabricantes criaram **Mnemônicos** (abreviaturas associadas com a função da instrução) para cada instrução, com o objetivo de facilitar a escrita de programas. Exemplo:

ADD A, 13h

"soma ao acumulador o valor armazenado na memória RAM no endereço 13h"

Opcode: **ADD A** **Operando:** o valor 13h

Opcode: 00100101 **Operando:** 00010011

- Programa em **Linguagem Assembly** é um programa contendo mnemônicos.

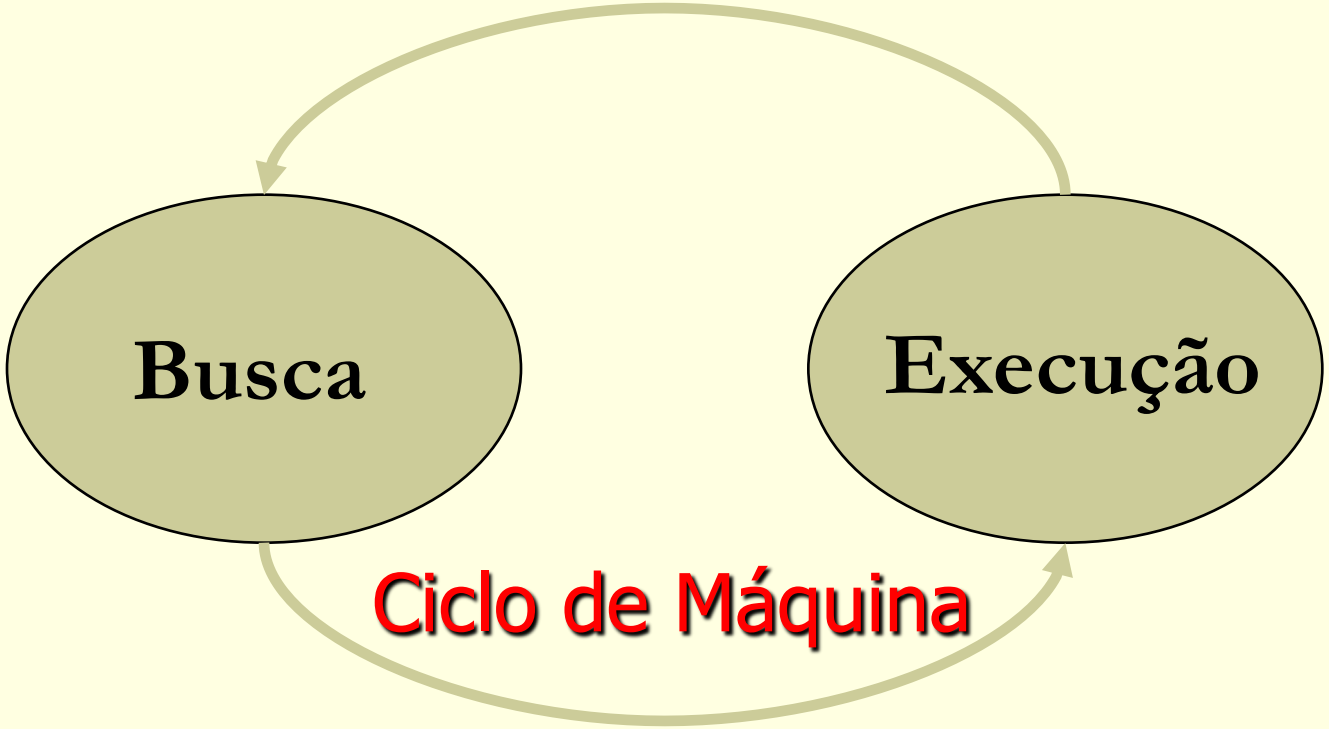
Programa em Linguagem Assembly

Como o Programa é inserido na memória ROM:

- O programador edita um programa em Linguagem Assembly (**Programa Fonte**), contendo instruções **(Mnemônicos)** do **μP**
- O programador usa um programa tradutor **(compilador)** para gerar o programa executável (**linguagem de máquina**)

- O programa objeto é carregado na ROM utilizando de um circuito programador **(PROM, EPROM, EEPROM, FLASH)**

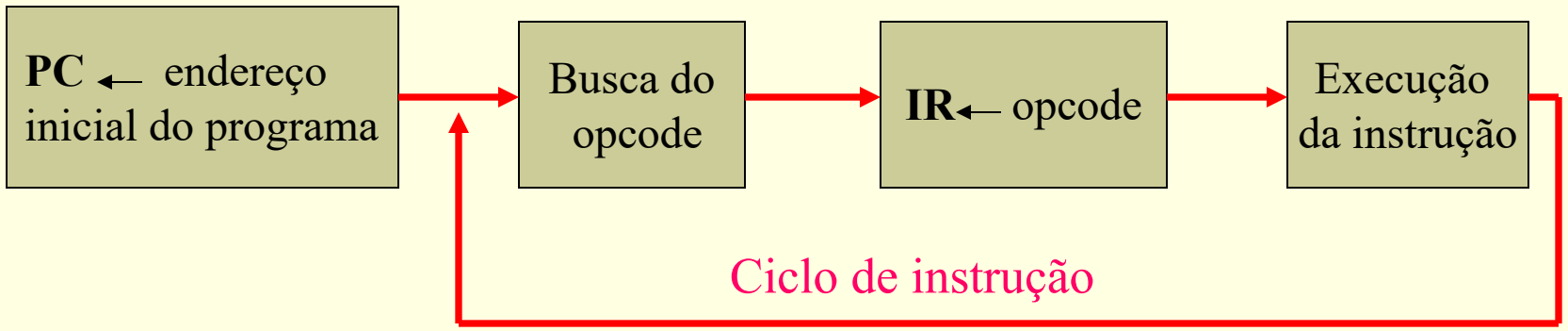
Ciclo de Máquina x Ciclo de Instrução



Ciclo de Instrução

- **Ciclo de Busca:** busca do opcode na memória de programa e armazena no IR;

- **Ciclo de Execução:** executa a instrução armazenada no IR ou busca mais bytes da mesma instrução se houver;



Existem instruções que gastam mais de um ciclo de máquina!

Ciclo de Máquina x Ciclo de Instrução

Exemplo para o 8051:

- Cada ciclo de máquina: ciclo de busca do "opcode" **+** leitura ou gravação, em memória ou I/O (duração de **12T**);
- Existem instruções de 1, 2 e 4 ciclos de máquina (até 48T);

Exemplo para o PIC:

- Cada ciclo de máquina: ciclo de busca do "opcode" **+** leitura ou gravação, em memória ou I/O (duração de **4T**);
- Existem instruções de 1 ou 2 ciclos de máquina;

FLAGS

Flags ➡ bits indicadores de estado da ULA:

- contidos no registrador PSW (palavra de status do programa – “program status word”)
- são setados ou limpados (1 ou 0) dependendo do resultado das operações da CPU
- algumas instruções testam flags para ver se elas devem ser executadas
- flags típicas: SIGN, CARRY, ZERO, OVERFLOW
- bit de flag usualmente se refere ao estado do A
- bit de sinal = MSB do A após a operação da ULA

Características das Instruções

Cada instrução é caracterizada por :

- Opcode
- Número de ciclos de máquina
- Número de períodos de clock (T)
- *Flag*s alterados pela instrução
- Modos de endereçamento

Características das Instruções

Microcontroller Instruction Set

For interrupt response time information, refer to the hardware description chapter.

Instructions that Affect Flag Settings(1)

Instruction	Flag			Instruction	Flag		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	O		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	O	X		ANL C,/bit	X		
DIV	O	X		ORL C,bit	X		
DA	X			ORL C,/bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

Note: 1. Operations on SFR byte address 208 or bit addresses 209-215 (that is, the PSW or bits in the PSW) also affect flag settings.

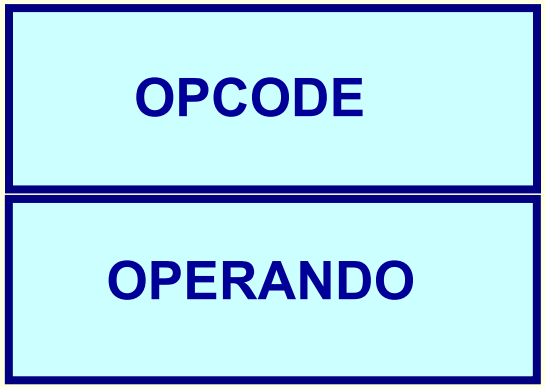
Características das Instruções

Table 1. AT89 Instruction Set Summary(1)

Mnemonic		Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS				
ADD	A,R_n	Add register to Accumulator	1	12
ADD	A,direct	Add direct byte to Accumulator	2	12
ADD	A,@R_i	Add indirect RAM to Accumulator	1	12
ADD	A,#data	Add immediate data to Accumulator	2	12
ADDC	A,R_n	Add register to Accumulator with Carry	1	12
ADDC	A,direct	Add direct byte to Accumulator with Carry	2	12
ADDC	A,@R_i	Add indirect RAM to Accumulator with Carry	1	12
ADDC	A,#data	Add immediate data to Acc with Carry	2	12
SUBB	A,R_n	Subtract Register from Acc with borrow	1	12
SUBB	A,direct	Subtract direct byte from Acc with borrow	2	12
SUBB	A,@R_i	Subtract indirect RAM from ACC with borrow	1	12
SUBB	A,#data	Subtract immediate data from Acc with borrow	2	12
INC	A	Increment Accumulator	1	12
INC	R_n	Increment register	1	12
INC	direct	Increment direct byte	2	12
INC	@R_i	Increment direct RAM	1	12
DEC	A	Decrement Accumulator	1	12
DEC	R_n	Decrement Register	1	12
DEC	direct	Decrement direct byte	2	12
DEC	@R_i	Decrement indirect RAM	1	12
INC	DPTR	Increment Data Pointer	1	24
MUL	AB	Multiply A & B	1	48
DIV	AB	Divide A by B	1	48
DA	A	Decimal Adjust Accumulator	1	12

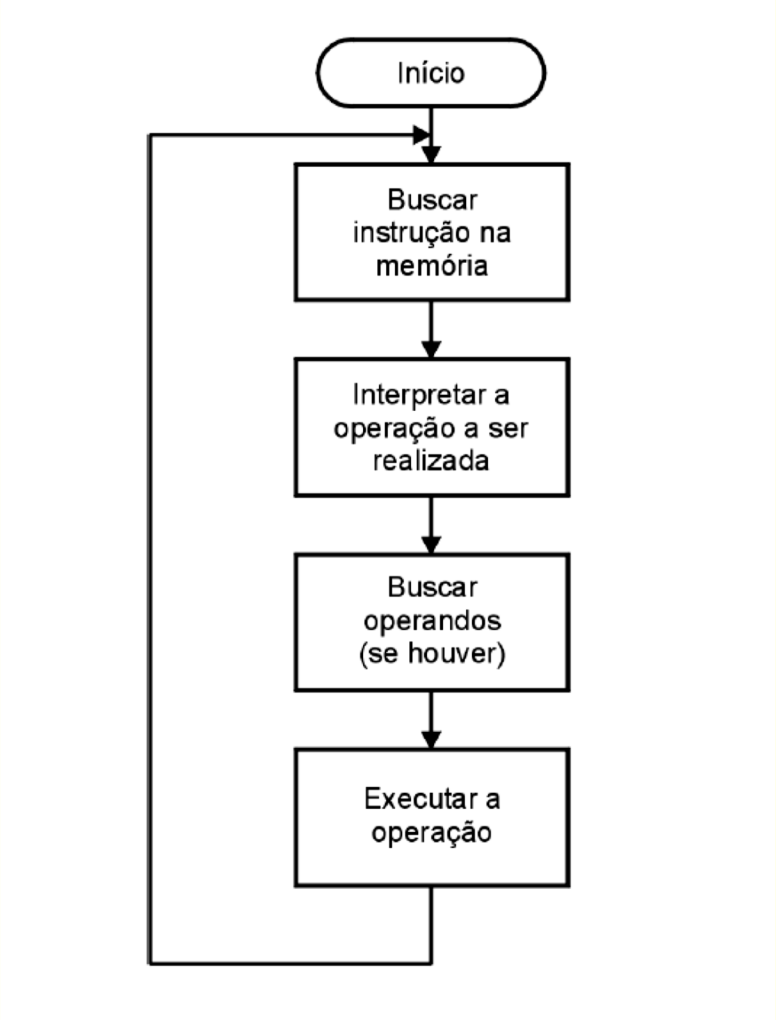
Classificação: Conjunto de Instruções

CISC - *Complex Instruction Set Computers*



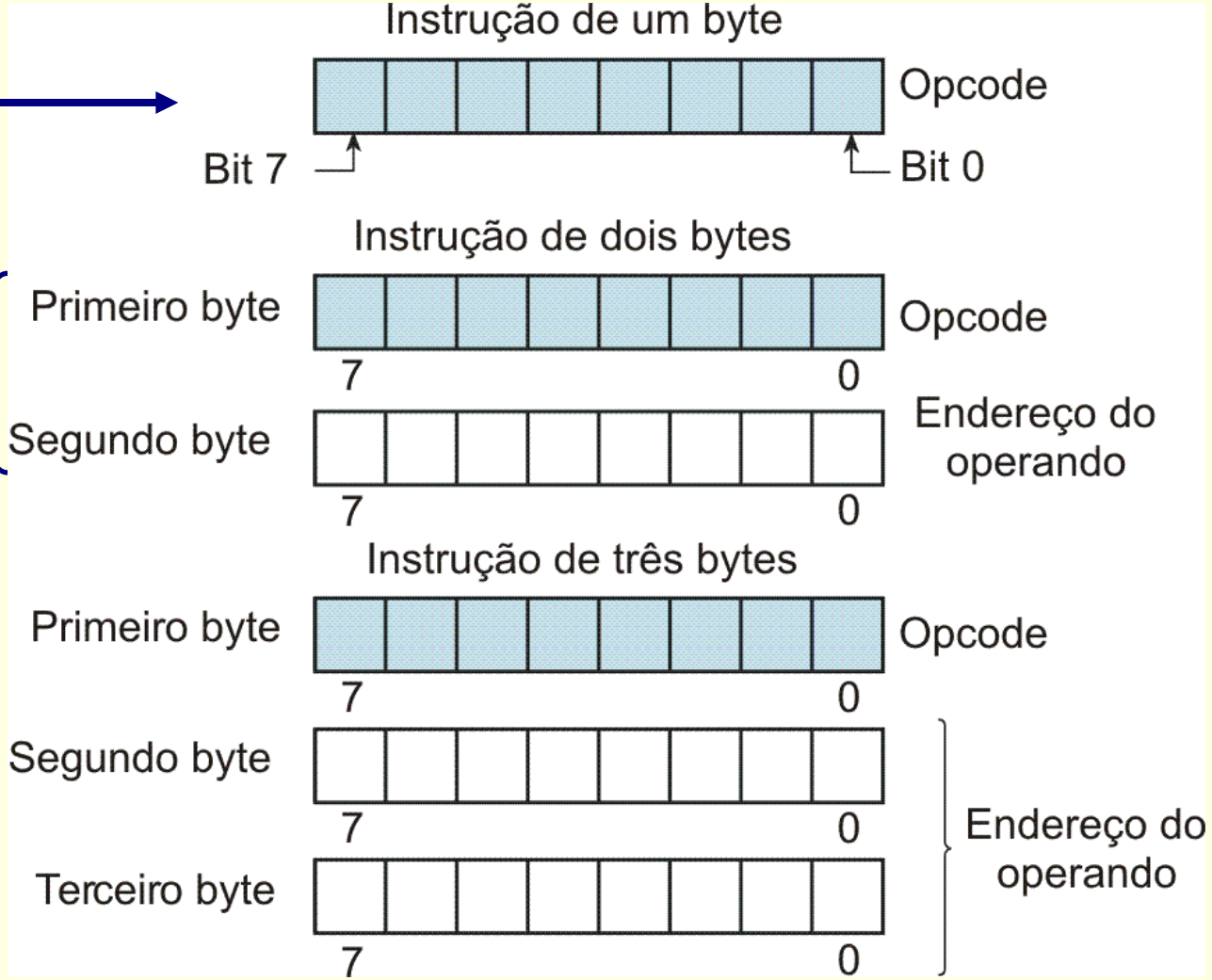
- *Opcode e Operando são armazenados em diferentes posições na memória ROM;*
- *Suporta um conjunto maior de instruções;*
- *Quantidade de instruções num programa geralmente é menor (mais instruções disponíveis ao programador);*
- *Processamento mais lento (instruções mais complexas que gastam mais de um ciclo de máquina);*

Ciclo de Instrução - CISC



Exemplos de Instruções CISC

CLR A



MOV A,30h

LJMP 3FB2h

Computadores CISC

Exemplo de um Programa Assembly do 8051

Memória

00	E5
01	30
02	B4
03	00
04	FB
05	80
06	FE

Endereço Conteúdo

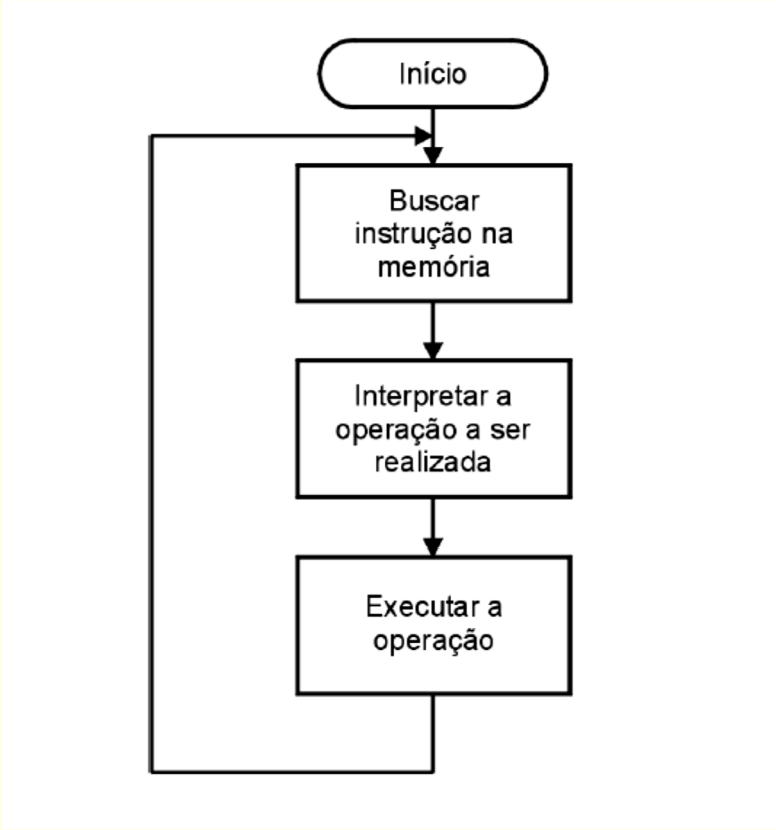
Addr	Opcodes	ASC	Label	Disassembly
0000	E5 30	å0	LOOP	MOV A,30h
0002	B4 00 FB	´		CJNE A,#00h,LOOP
0005	80 FE	€þ	AQUI	SJMP AQUI

Classificação: Conjunto de Instruções

RISC - *Reduced Instruction Set Computers*

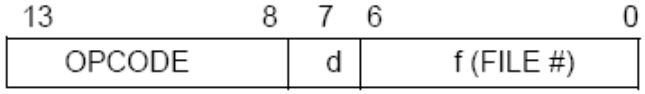
- *Opcode e Operando são armazenados na mesma posição na memória ROM;*
- *Implementa quantidade limitada de instruções ⇨ otimizadas ⇨ maior rapidez*
- *Gastam em sua maioria apenas um ciclo de máquina*
- *Todas as instruções ocupam o mesmo tamanho na memória*
- *Programa ⇨ quantidade maior de instruções ⇨ menos instruções disponíveis ao programador.*

Ciclo de Instrução - RISC



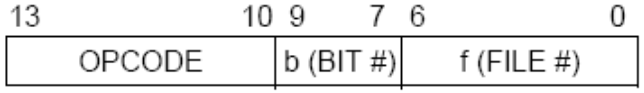
Exemplos de Instruções RISC

Byte-oriented file register operations



d = 0 for destination W
d = 1 for destination f
f = 7-bit file register address

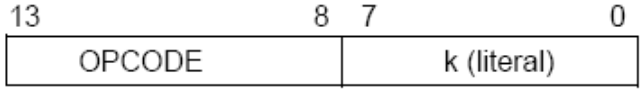
Bit-oriented file register operations



b = 3-bit bit address
f = 7-bit file register address

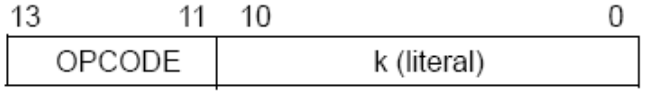
Literal and control operations

General

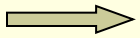


k = 8-bit immediate value

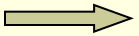
CALL and GOTO instructions only



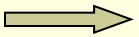
k = 11-bit immediate value



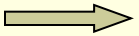
MOVF STATUS, W



BCF STATUS, RP0



MOVLW B‘00011100’



CALL SUBROTINA

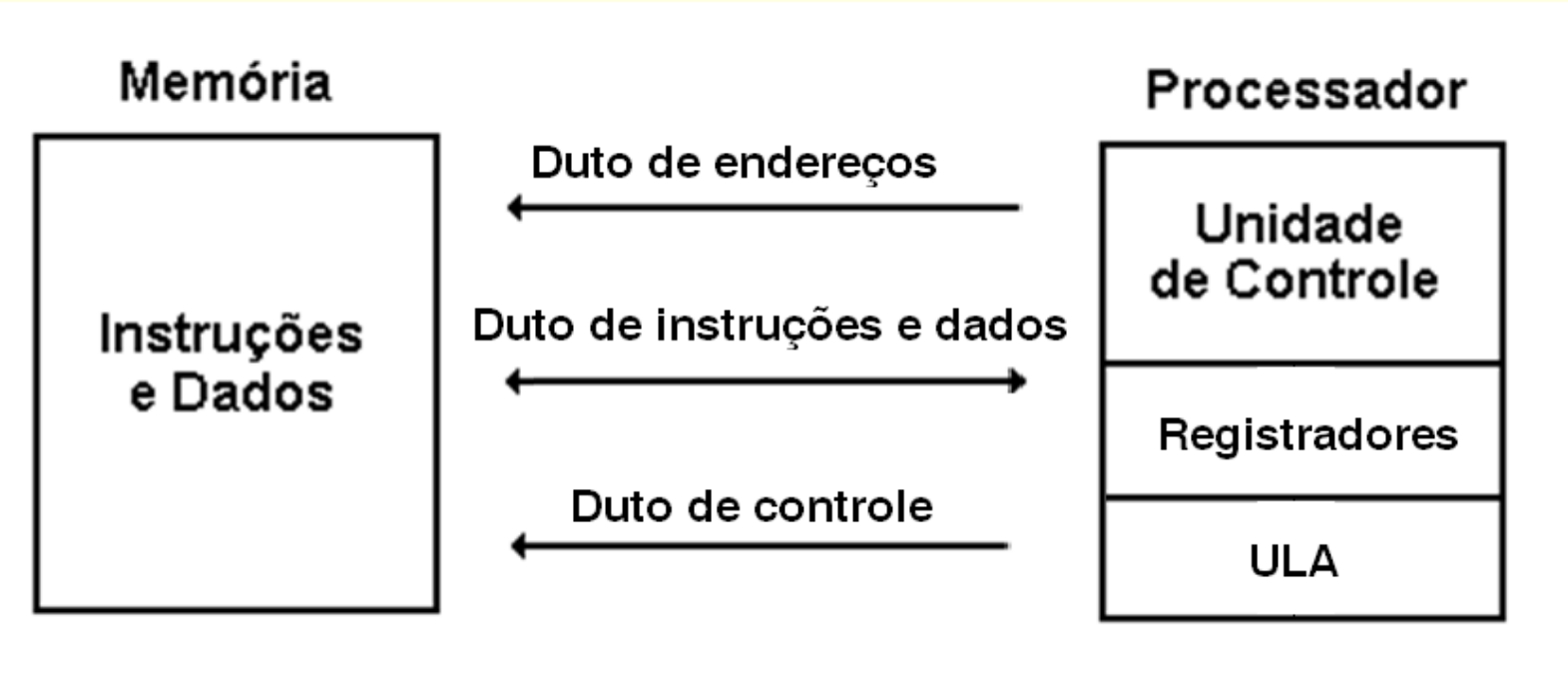
RISC x CISC

- **CISC:**
  - Instruções podem ocupar espaços diferentes na memória de programa (Opcode + operando)
  - Acabam tendo durações diferentes;
  - Ocupam mais espaço na memória de programa;
  - Mais instruções disponíveis = programa mais simples.
- **RISC:**
  - Cada instrução ocupa o mesmo espaço na memória de programa (Opcode + operando);
  - Todas tem a mesma duração (exceto as de "salto");
  - Ocupam menos espaço na memória de programa;
  - Menos instruções disponíveis = programas mais complexos.

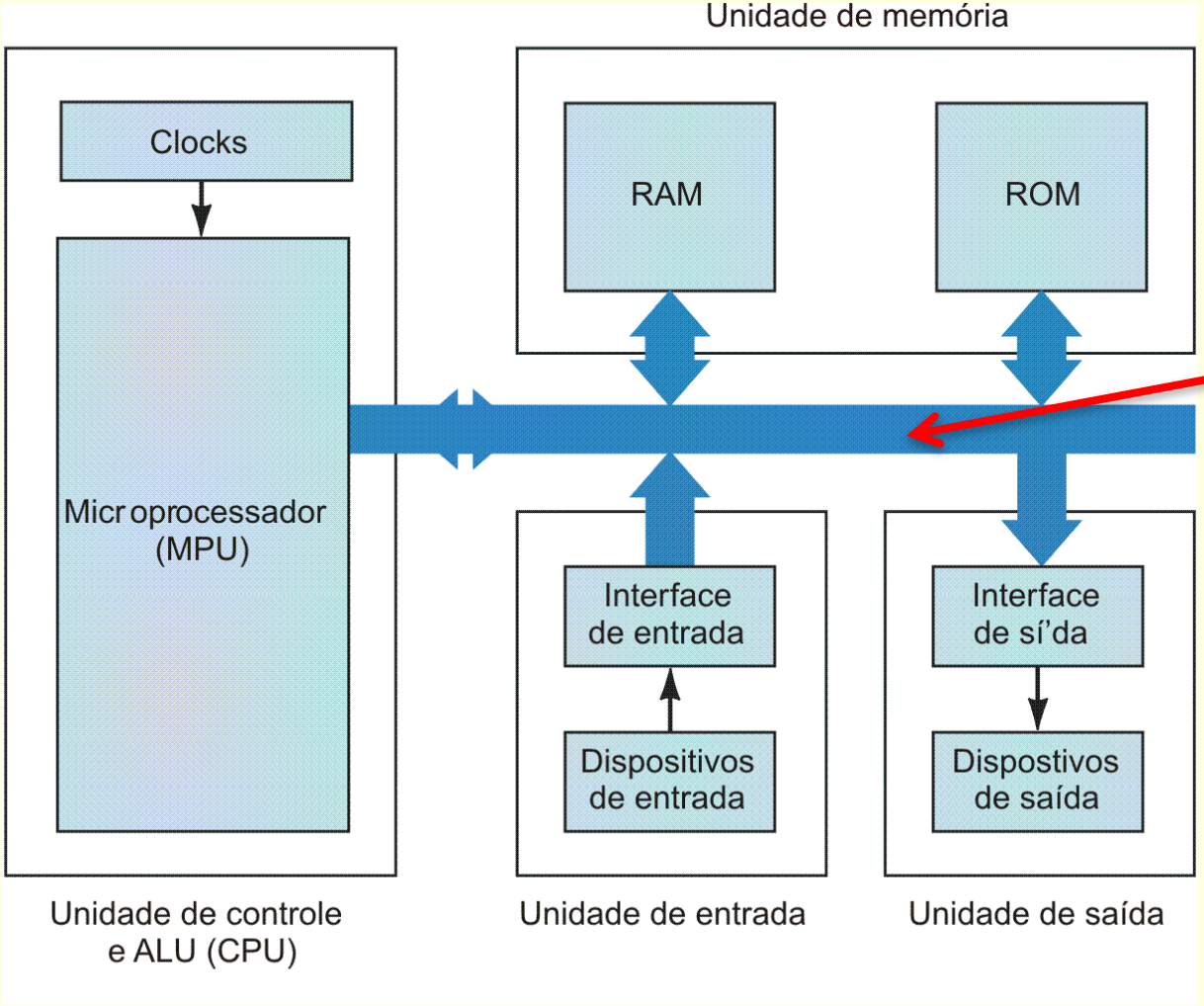
Modelos de Arquiteturas para Computadores

Arquitetura de Von Neumann
X
Arquitetura Harvard

Arquitetura Von Neumann

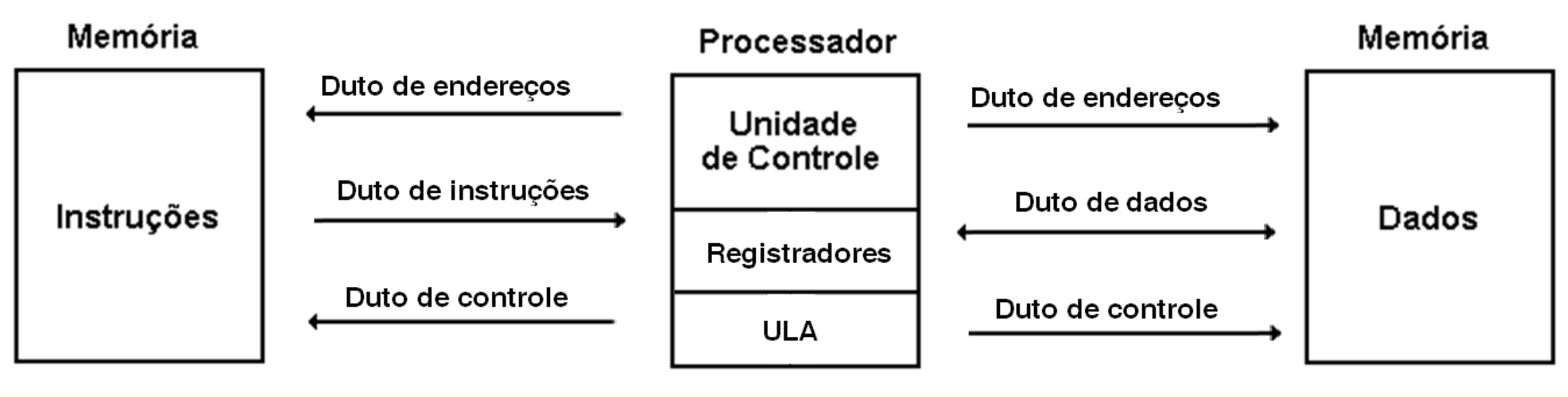


Microcontrolador Intel 8051
Arquitetura Von Neumann

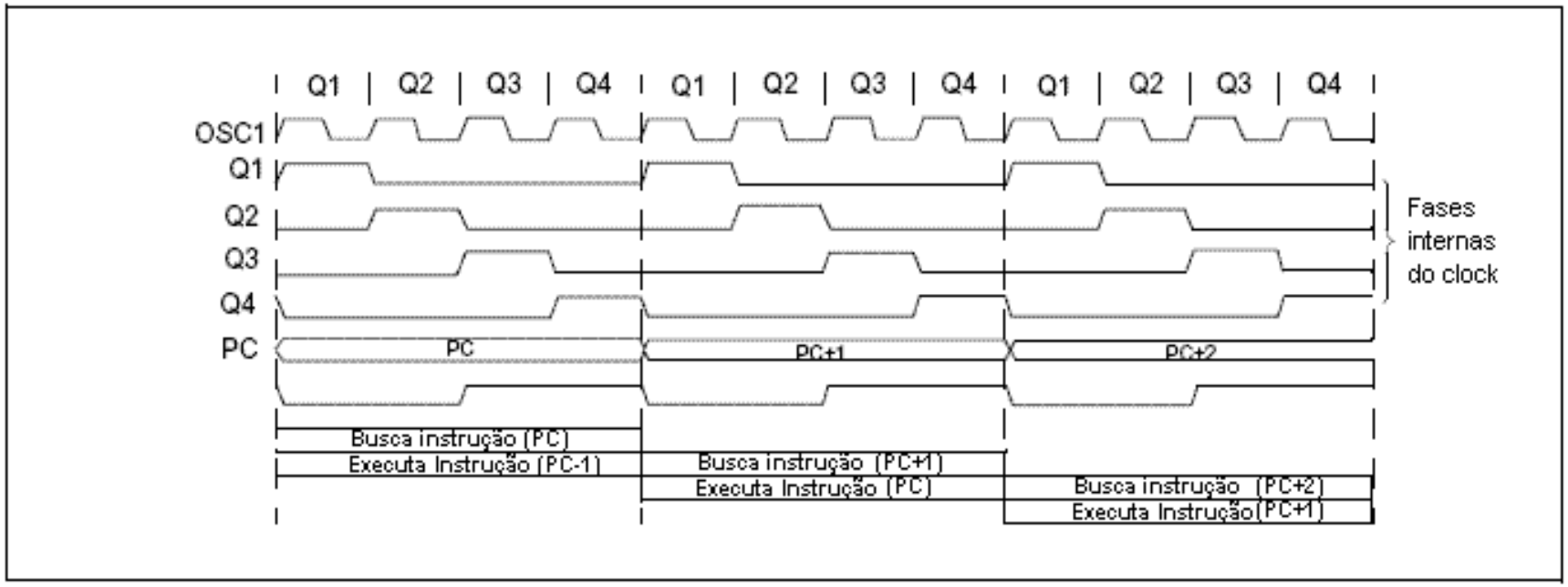


Apesar de duas memórias, elas compartilham o mesmo barramento

Arquitetura Harvard

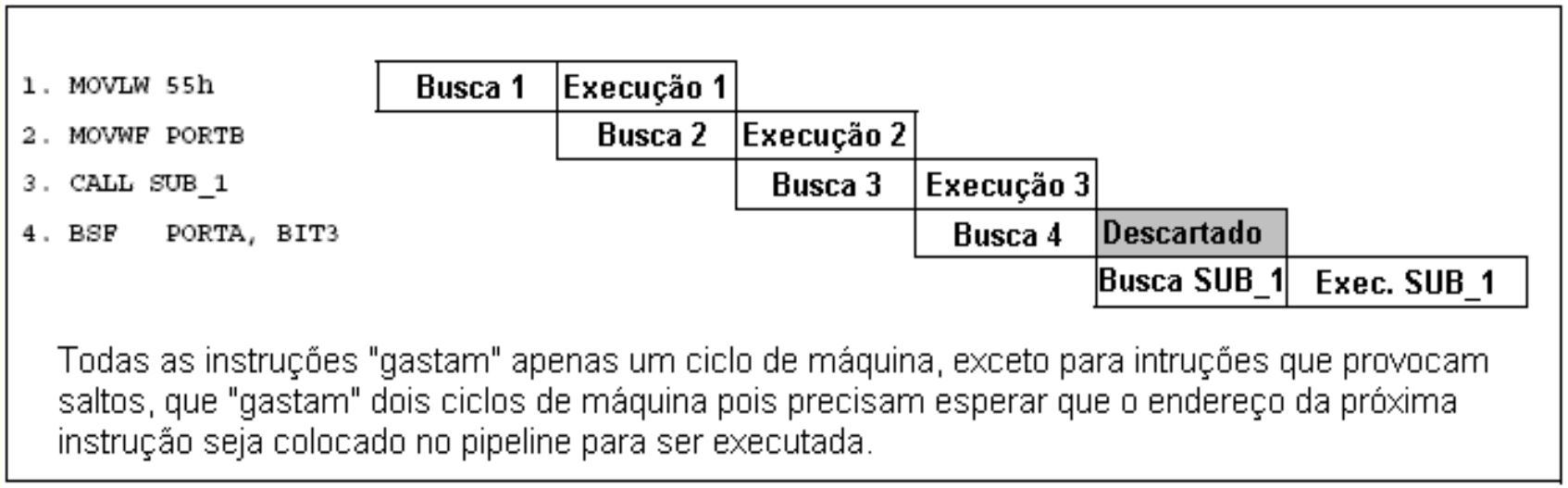


Pipelining de 2 estágios: µcontroladores PIC



- Ciclo de máquina = fosc/4 para os µcontroladores PIC
- As instruções devem ser de um ciclo de máquina

Pipelining de 2 estágios: µcontroladores PIC



- Busca e execução em apenas 1 ciclo de máquina;
- Instruções de "salto" gastam dois ciclos de máquina;

Arquitetura Harvard

- Busca e execução em apenas 1 ciclo de máquina;
- Todas as instruções tem largura fixa;
- Poucas instruções gastam mais de 1 ciclo de máquina;
- Pipelining de 2 ou mais estágios:
  - Intel 8086, 8088, PIC: 2 estágios
  - Pentium: 5 estágios
  - Pentium 4: 20 estágios

Von Neumann X Harvard

- **Von Neumann**:
  - Arquitetura mais simples;
  - Mais lento pois não permite acesso simultâneo às memórias;
  - Geralmente CISC

Exemplo:

4004	– 46 instruções
8080	– 78 instruções
8085	– 150 instruções
Z80	– Mais de 500 instruções

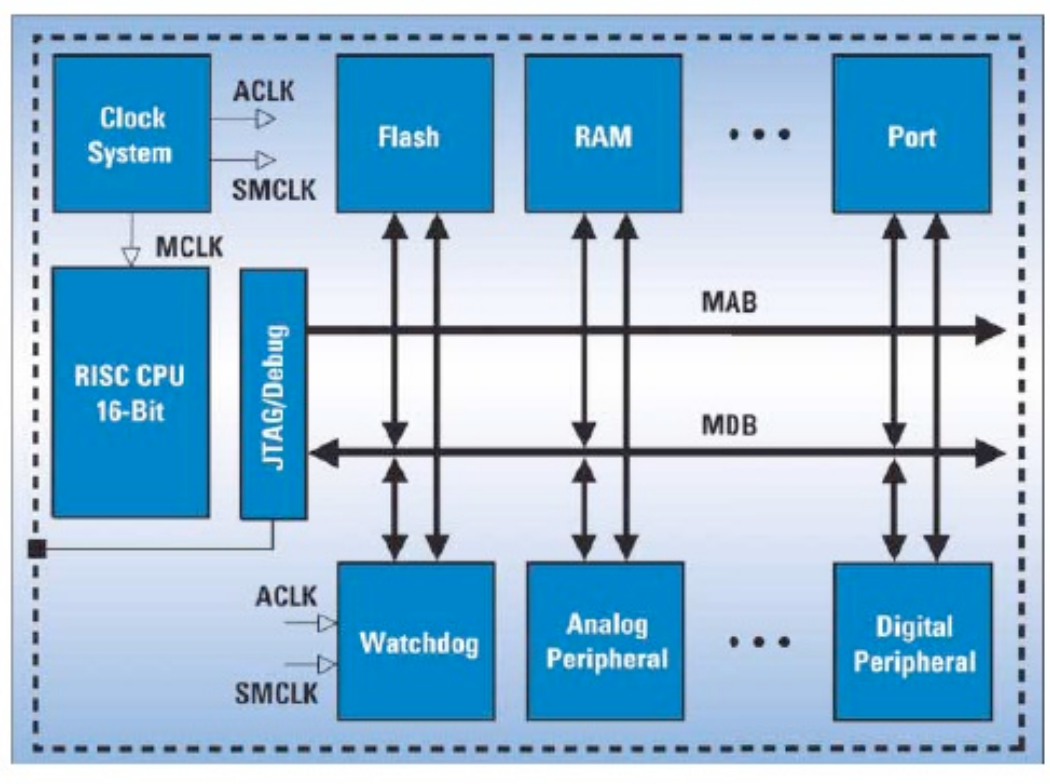
Von Neumann X Harvard

- **Harvard:**
  - Arquitetura mais complexa;
  - Mais rápido, pois permite acesso simultâneo às memórias;
  - Geralmente RISC
  - Permite o Pipelining

  Exemplo:
  - Intel 8086, 8088
  - Microchip PIC – 35 instruções

Arquitetura Von Neummann com Set de Instruções RISC

- **Texas MSP430:**
  - Arquitetura Von Neumann;
  - Intruções RISC de 16 bits;



FIM