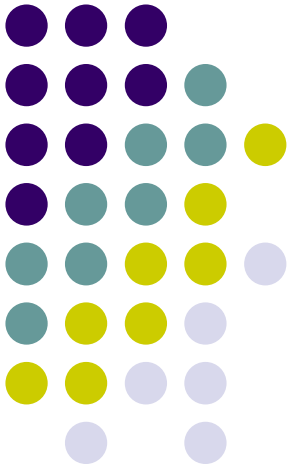
Organização de Computadores

Conceito de Programa
Ciclo de Instrução

Prof. José Paulo G. de Oliveira
Engenharia da Computação, UPE
jpgo@ecomp.poli.br



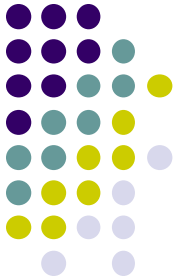
Índice

- Conceito de Programa
- Função da Unidade de Controle
- Ciclo de instrução básico
- Interrupções
- Múltiplas Interrupções

"I want a computer that does what I want it to do, not what I tell it to do!"

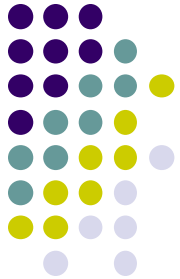
Conceito de Programa

Motivação



- Sistemas de hardware dedicado são inflexíveis

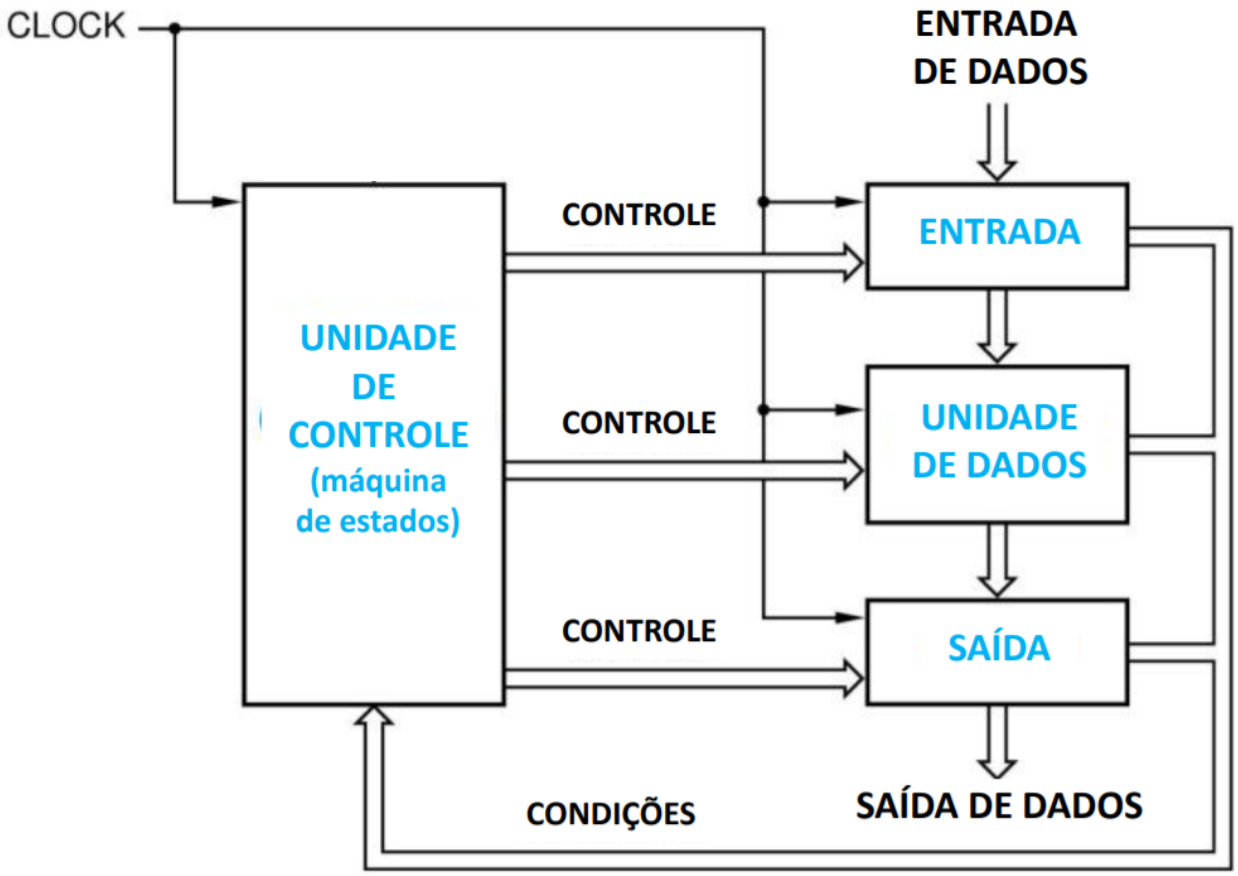
Motivação



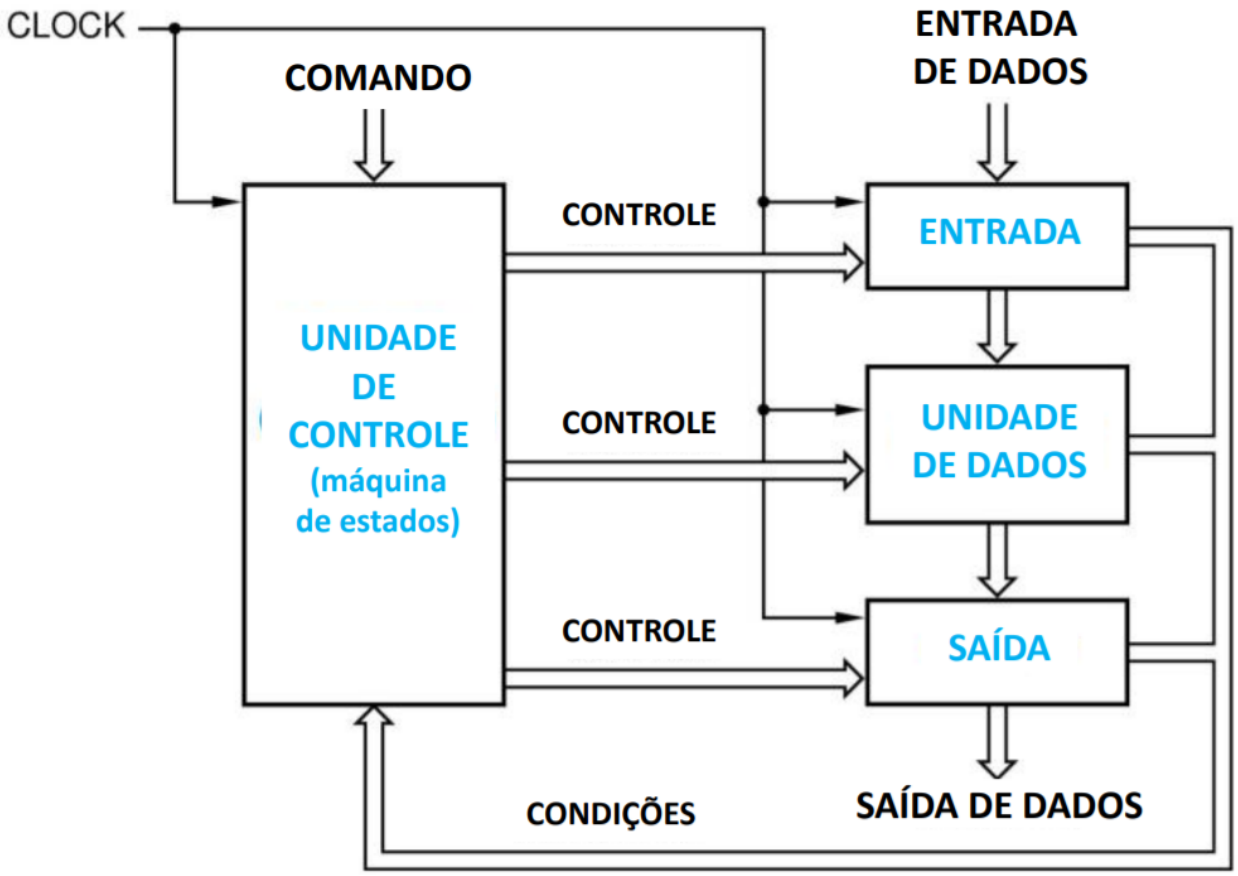
- Sistemas de hardware dedicado são inflexíveis

- Em contrapartida...um hardware de propósito geral pode executar diferentes tarefas
  - Fornecidos os sinais de controle adequados
  - Em vez de fazer novos circuitos, é fornecido um novo conjunto de sinais de controle

Sistema Digital Dedicado

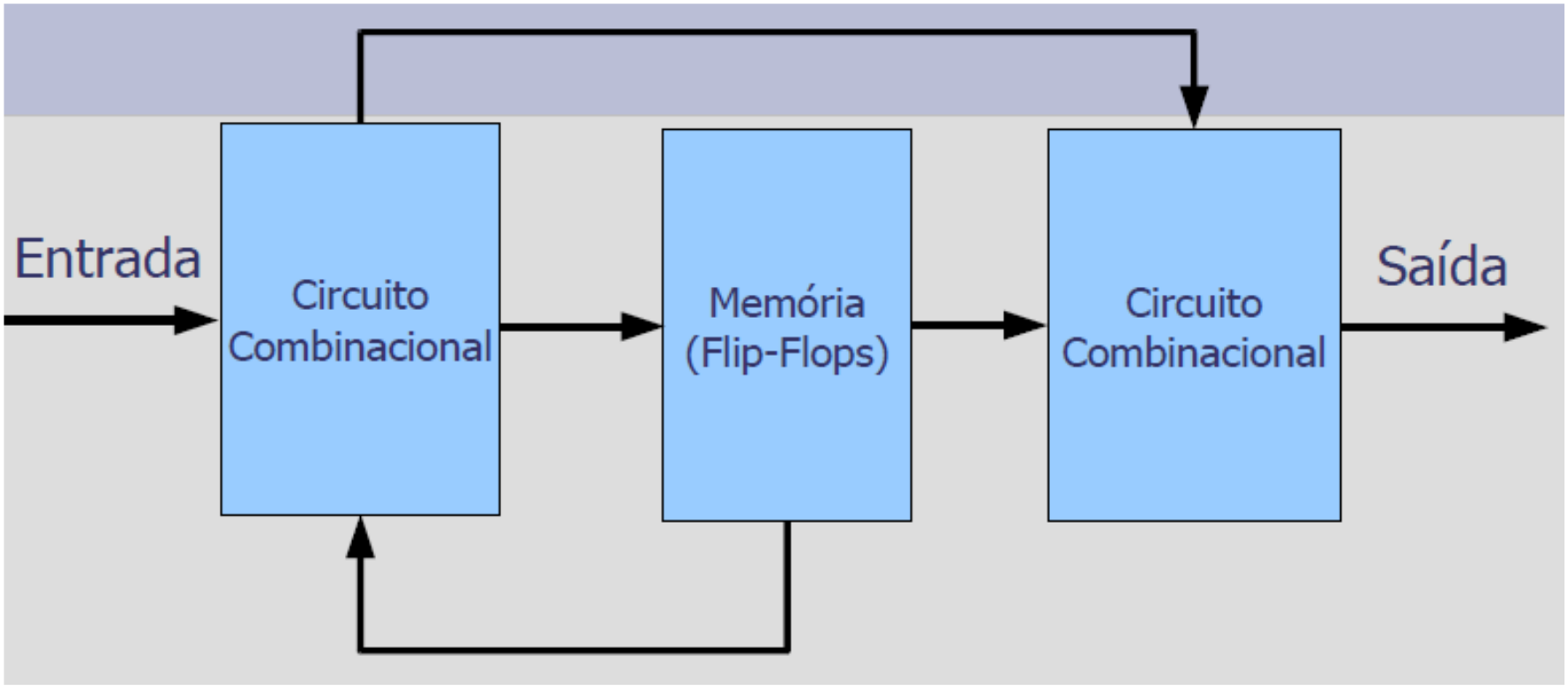


Sistema Digital Flexível



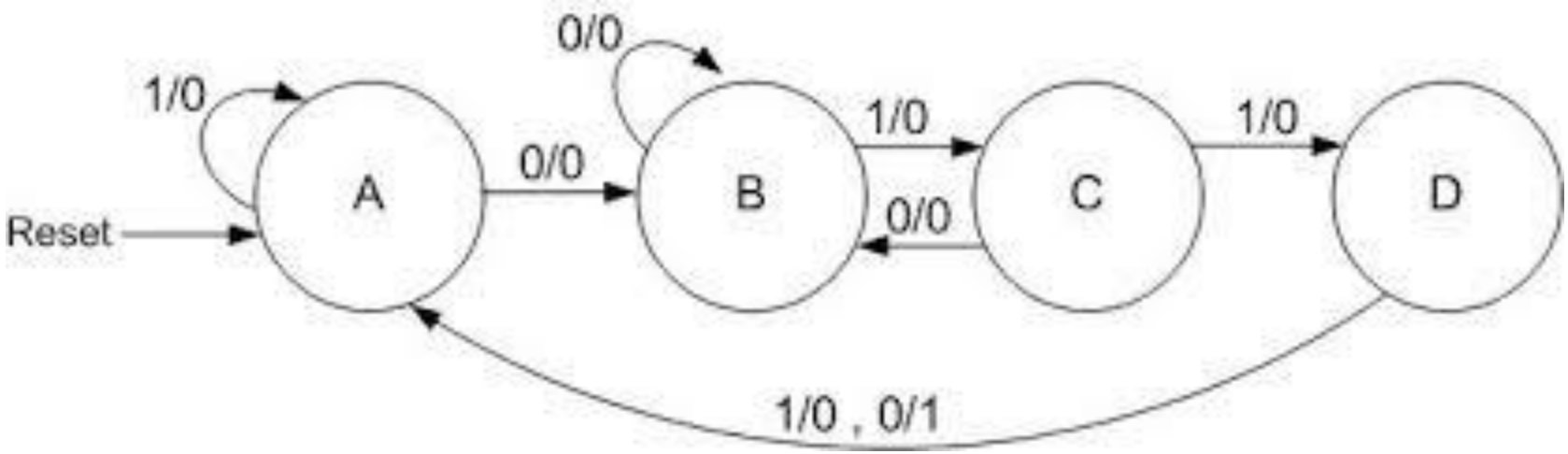
Máquina de Estados Finitos

Ex.: Máquina Mealy

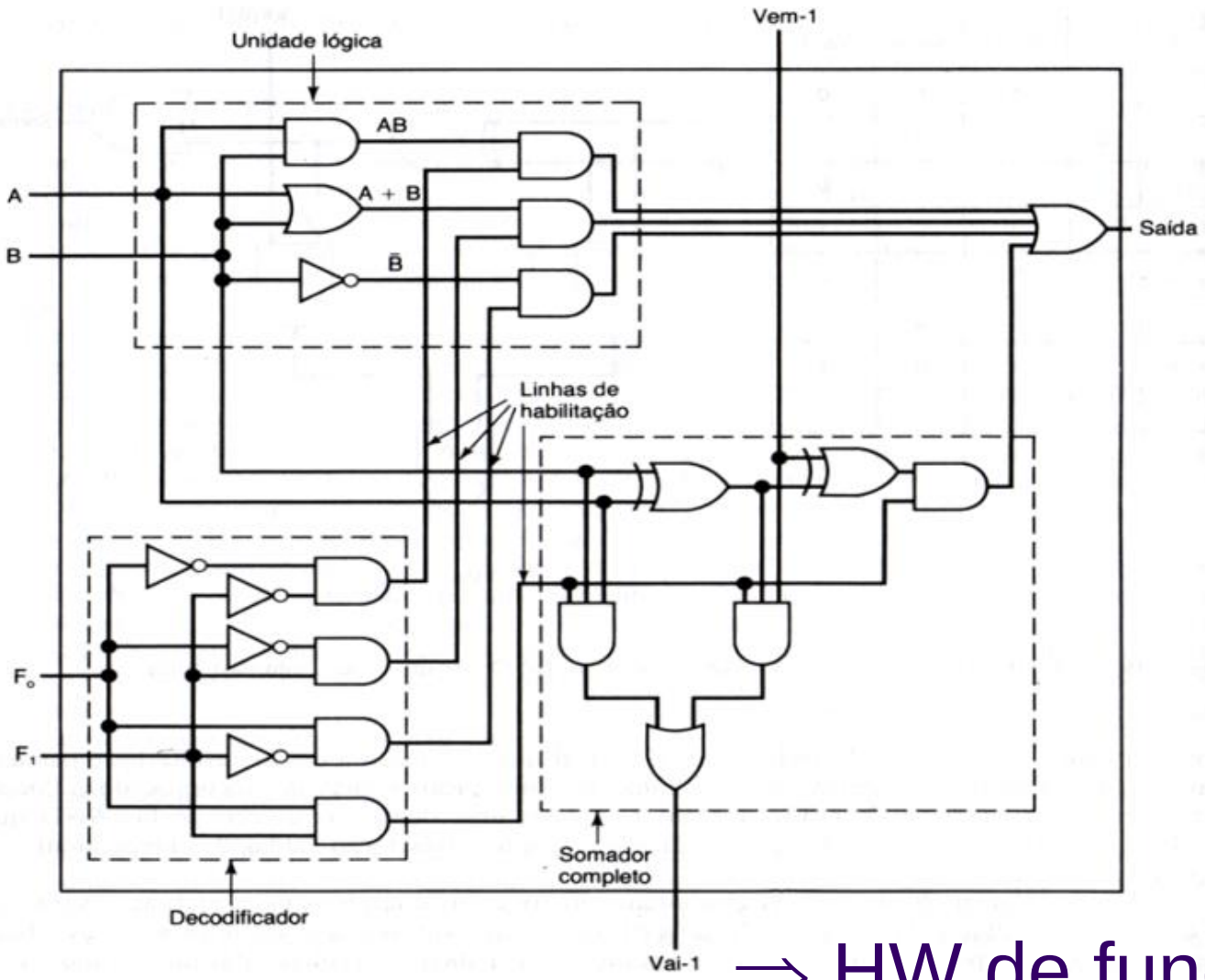


$\Rightarrow$ HW Dedicado

Ex.: Máquina Mealy

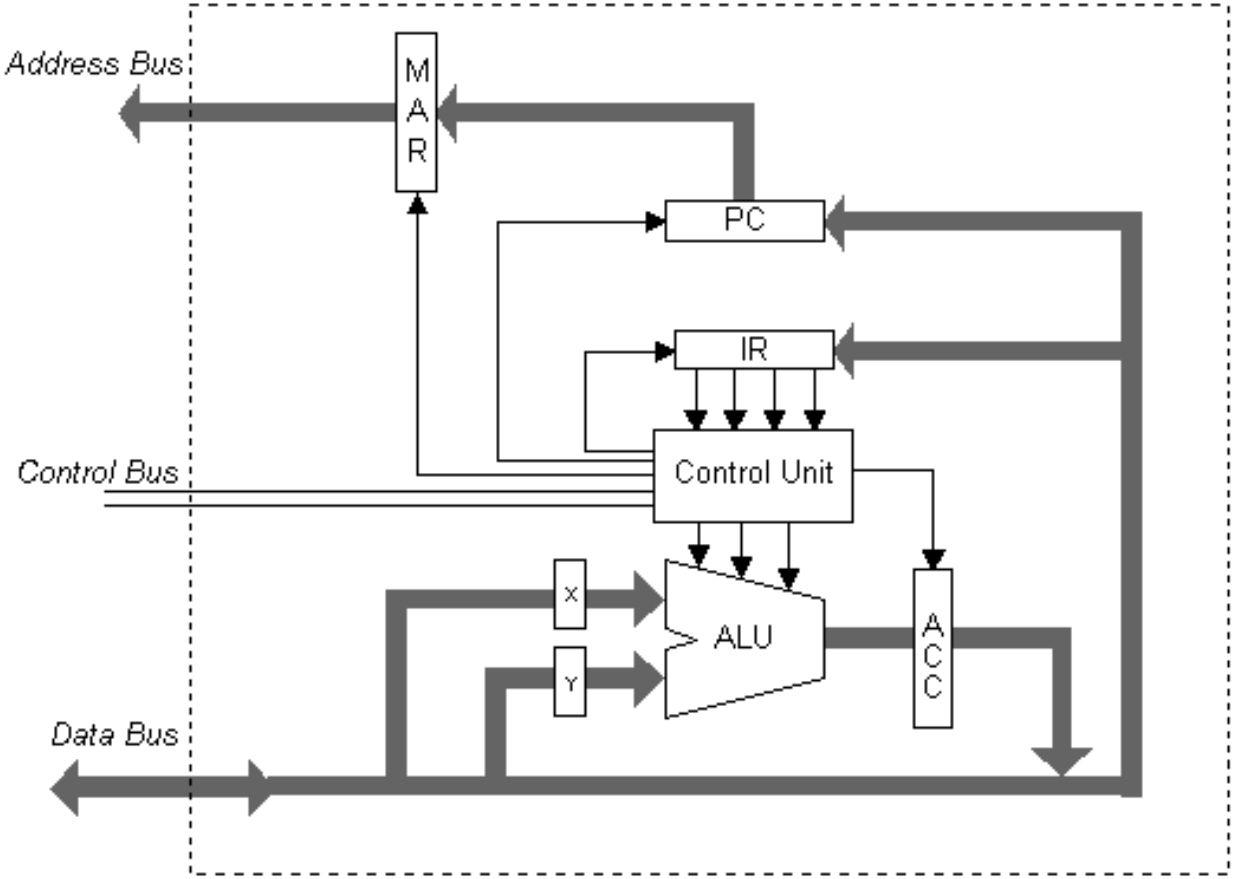


Ex.: ULA*

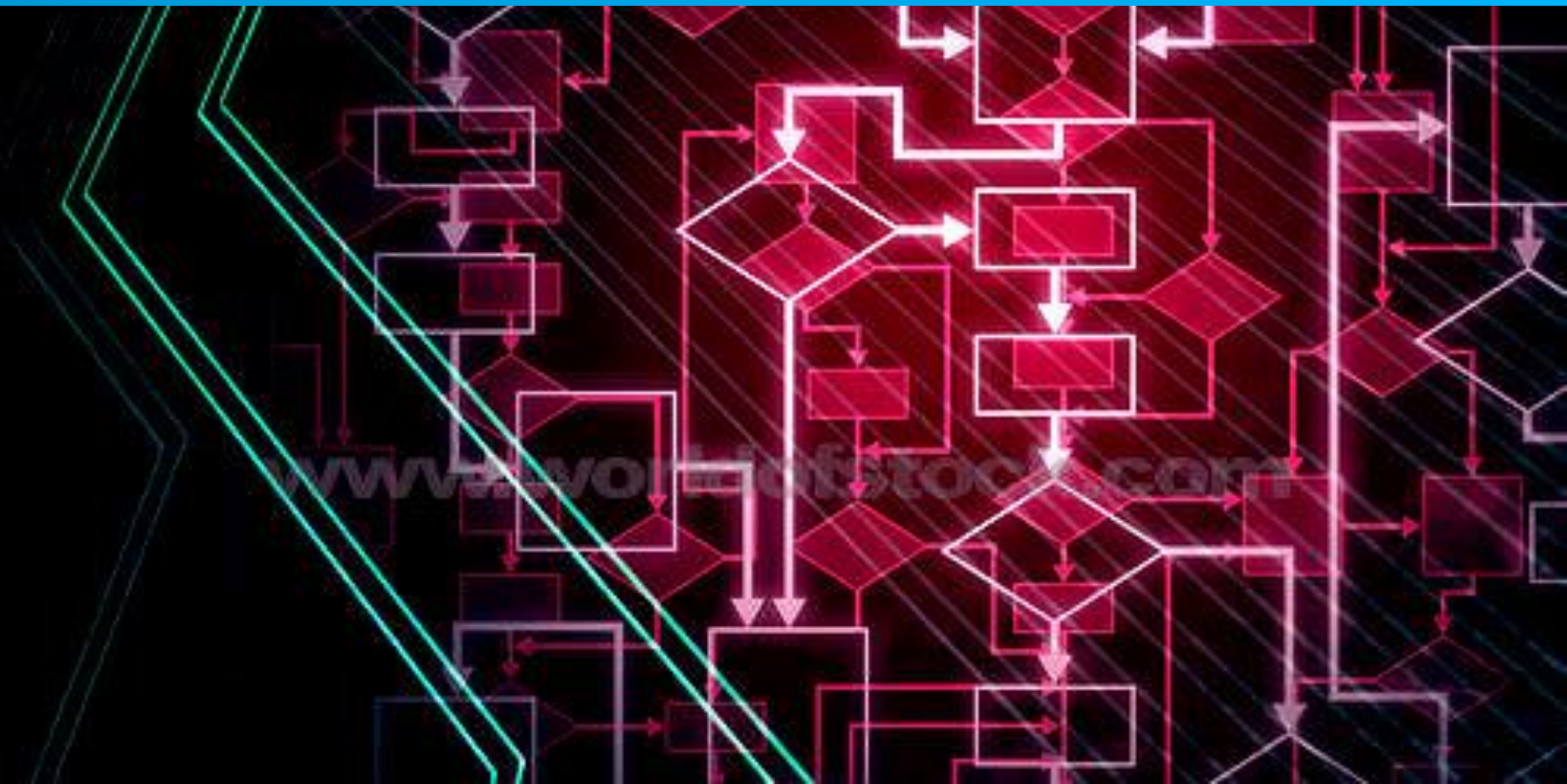


$\Rightarrow$ HW de função ajustável

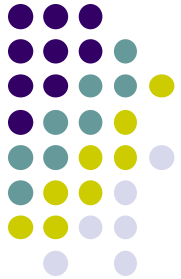
CPU


Address Bus
M
A
R
PC
IR
Control Unit
Control Bus
X
Y
ALU
A
C
C
Data Bus

O que é um Programa de Computador?



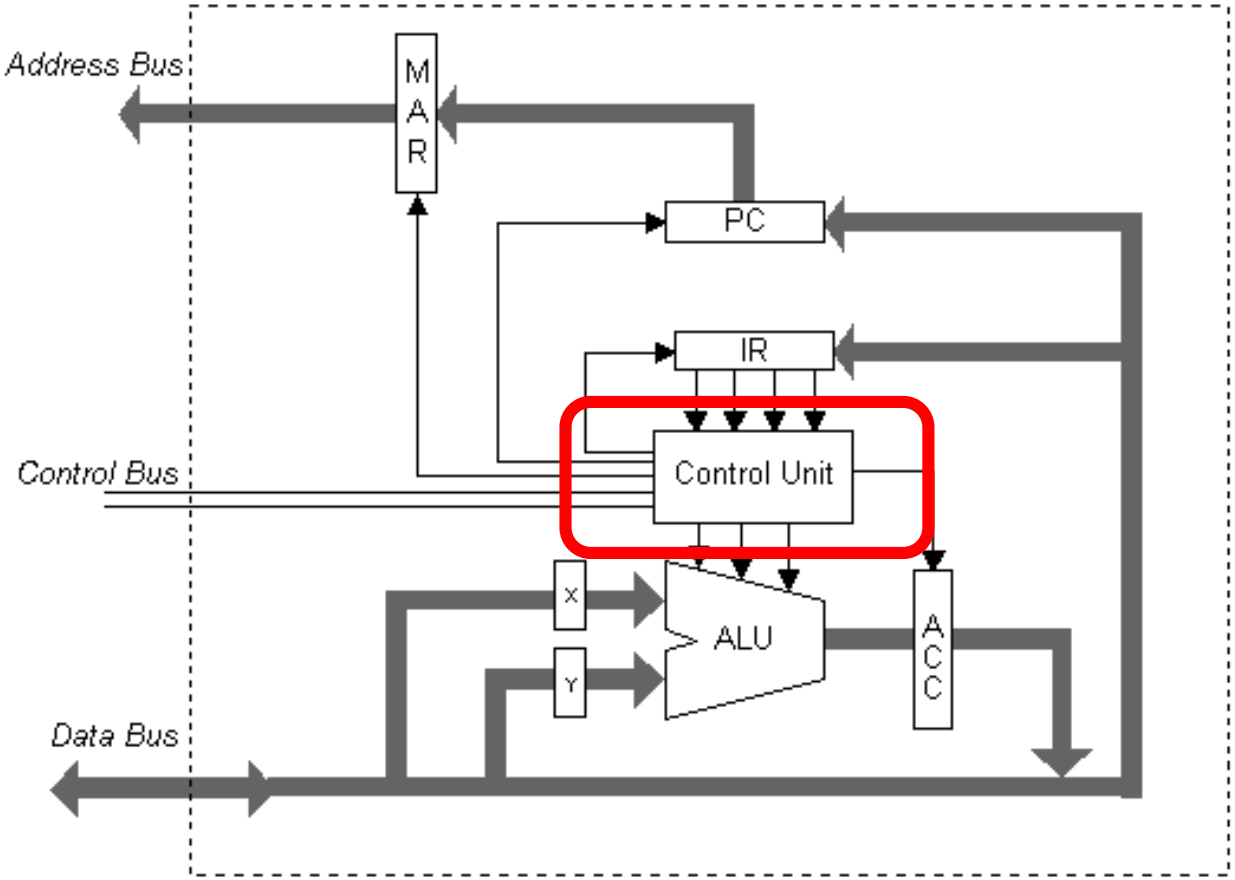
O que é um programa?



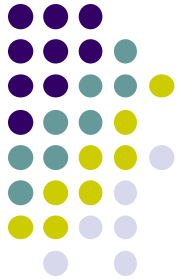
- Uma sequência de passos
- A cada passo uma operação lógica, aritmética ou de transferência é realizada
- Um conjunto de sinais de controle diferente é necessário para cada operação

Unidade de Controle

Unidade de Controle



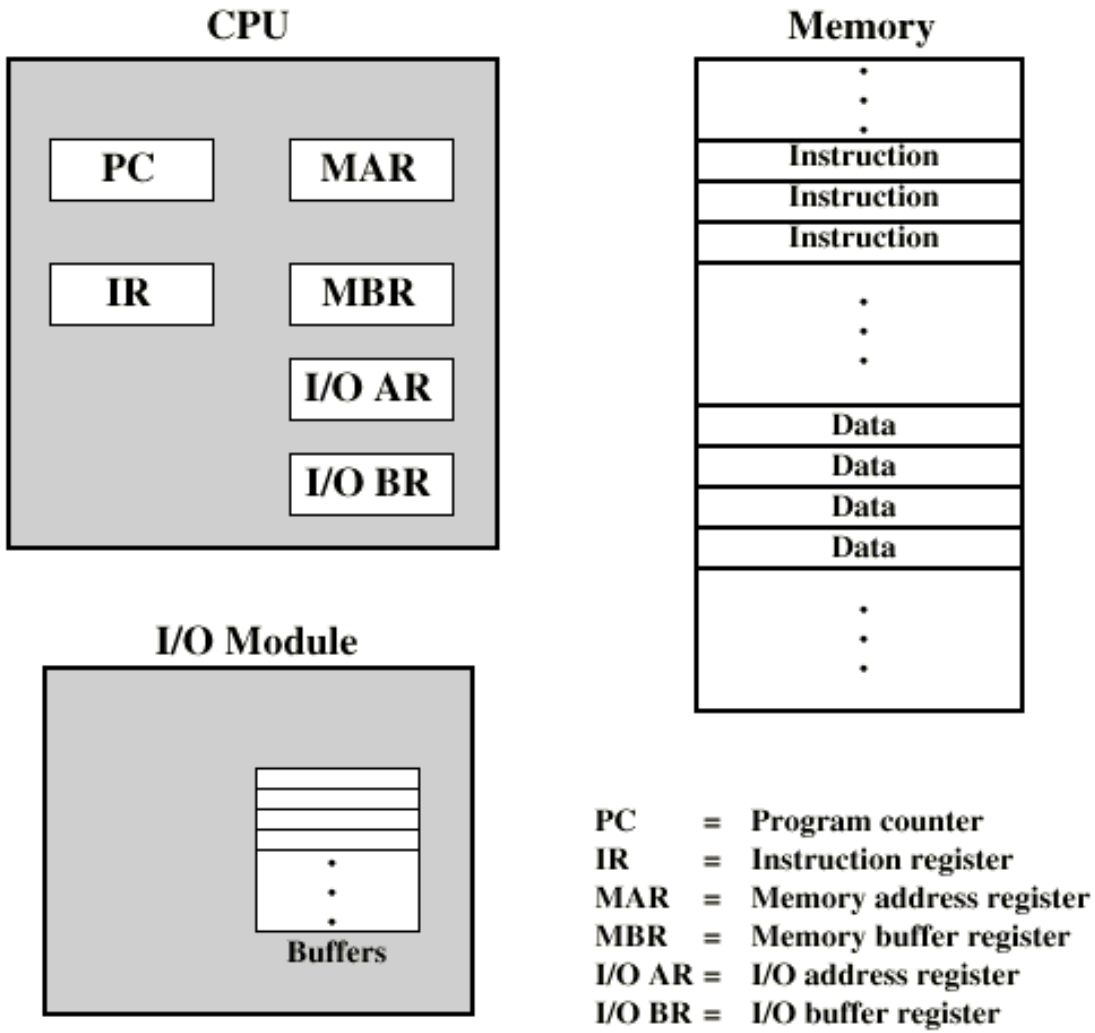
Função da Unidade de Controle



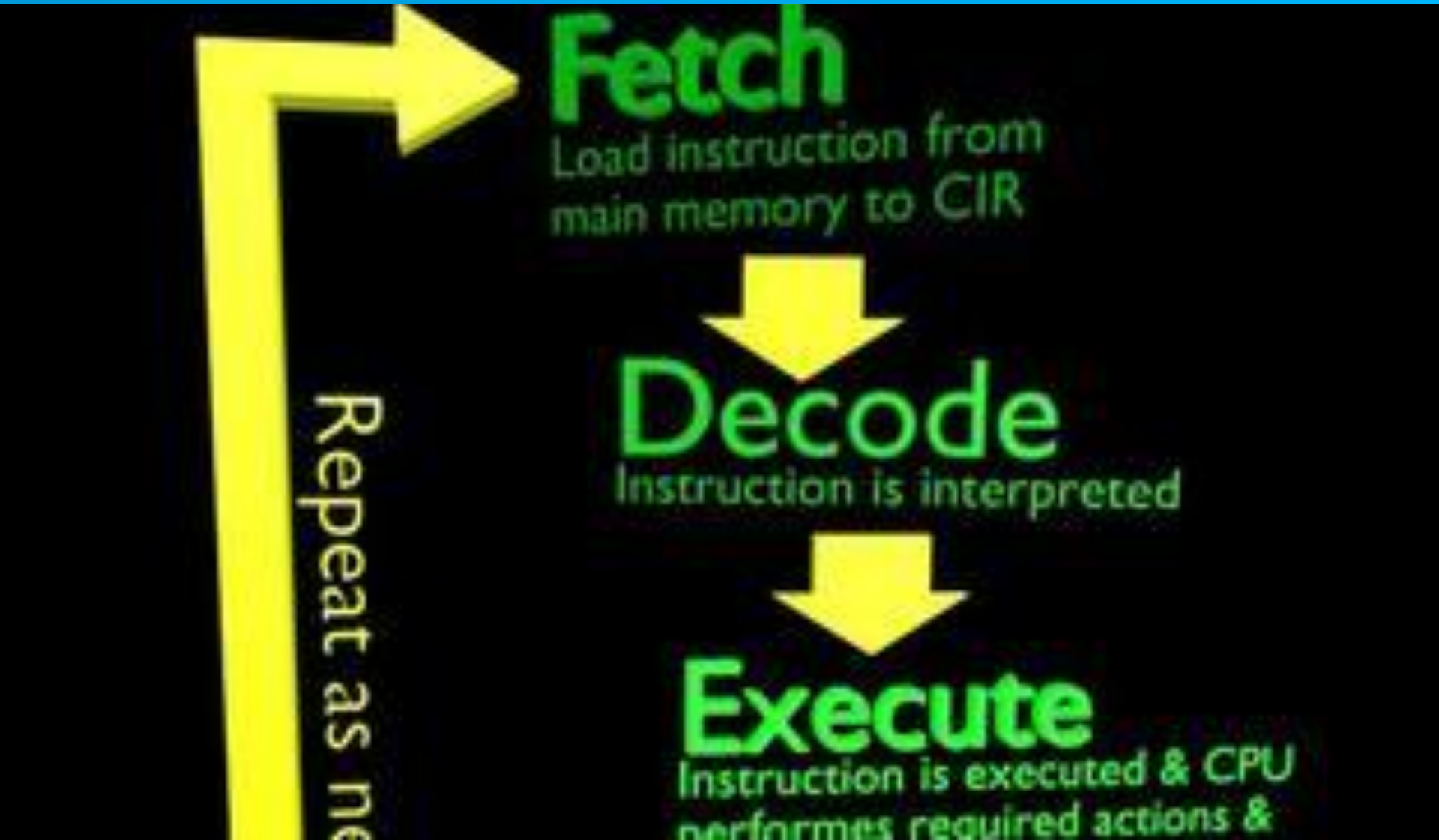
- Para cada operação, um código único é fornecido
  - Ex.: ADD, MOVE
- A unidade de controle recebe o código e gera os sinais adequados
- Os módulos de hardware respondem aos sinais de controle

- Eis um computador!

Componentes do Computador: Alto Nível

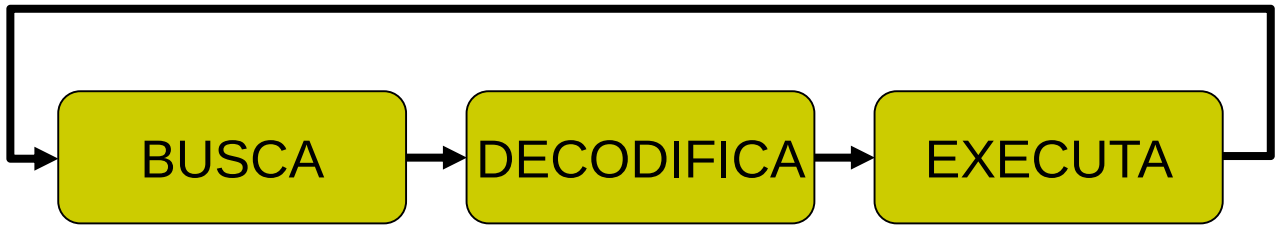


Ciclo de Instrução

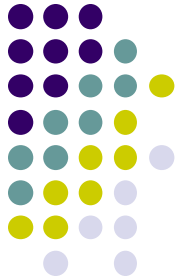


Ciclo de instrução básico

- Três passos:
  - Busca
  - Decodificação
  - Execução

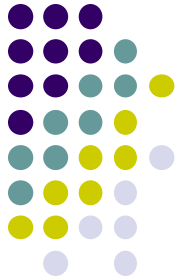


Ciclos de Busca/Decodificação



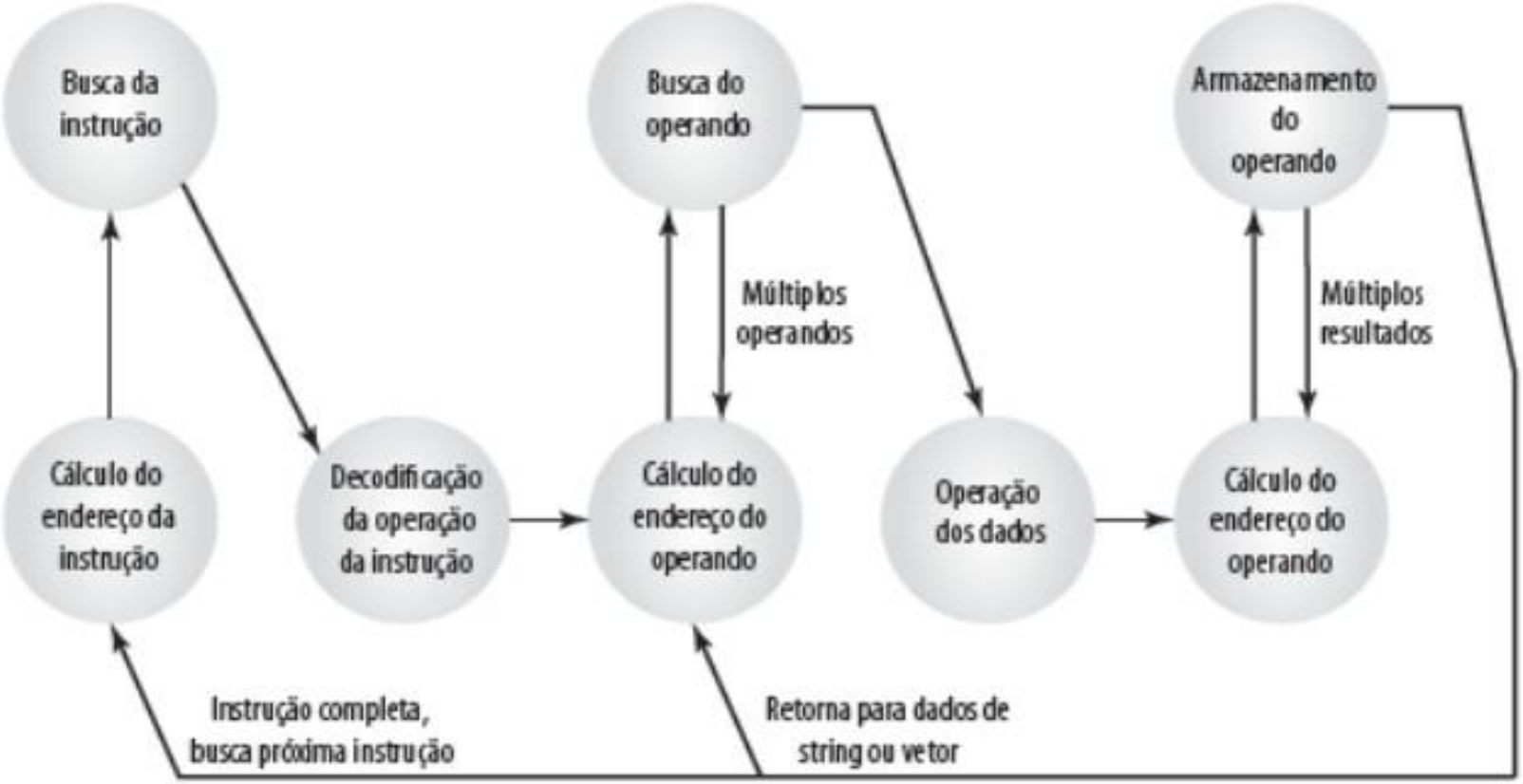
- O Contador de Programa (PC) armazena o endereço da próxima instrução a ser buscada
- O processador busca a instrução na posição de memória apontada pelo PC
- O PC é incrementado
  - A não ser que ocorra um desvio
- A instrução é carregada no Registrador de Instrução (IR)
- O processador interpreta a instrução e realiza as ações requisitadas

Tipos de Operações

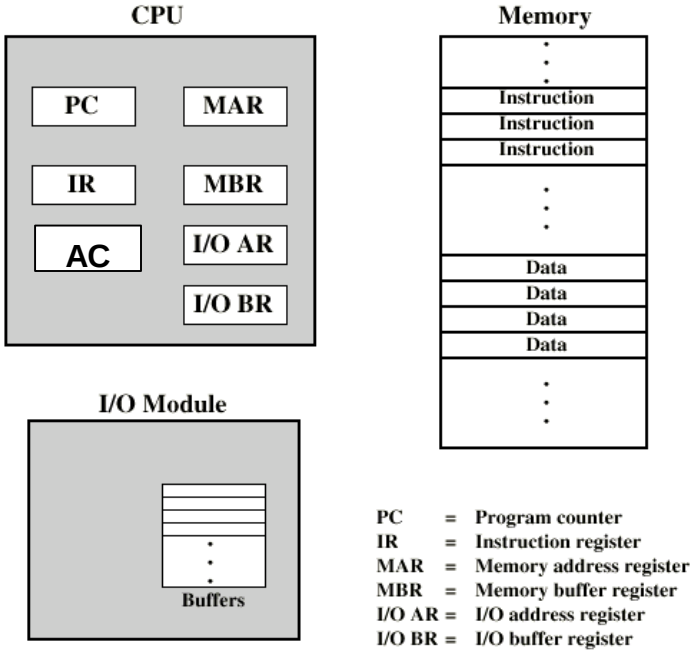


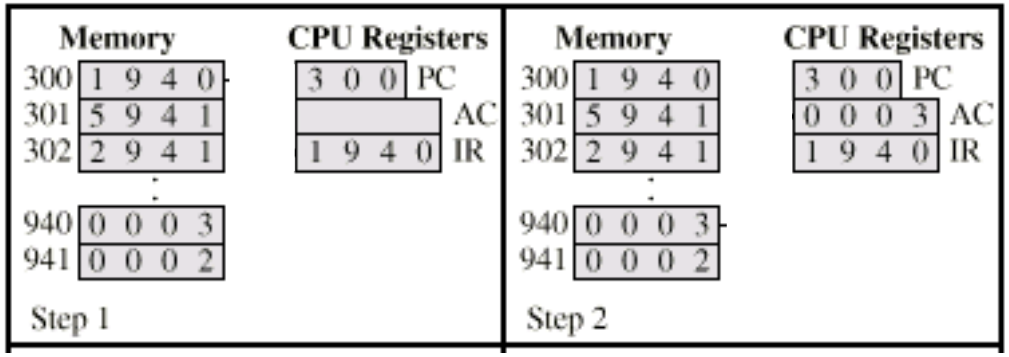
- Processador-memória
  - Transferência de dados entre a CPU e a memória principal
- Processador-E/S
  - Transferência de dados entre a CPU e os módulos de E/S
- Processamento de dados
  - Operações lógicas ou aritméticas sobre os dados
- Controle
  - Alteração da sequência de operações
  - Ex.: JMP; JE
- Combinação de todas acima

Ciclo de Instrução – Diagrama de Estados

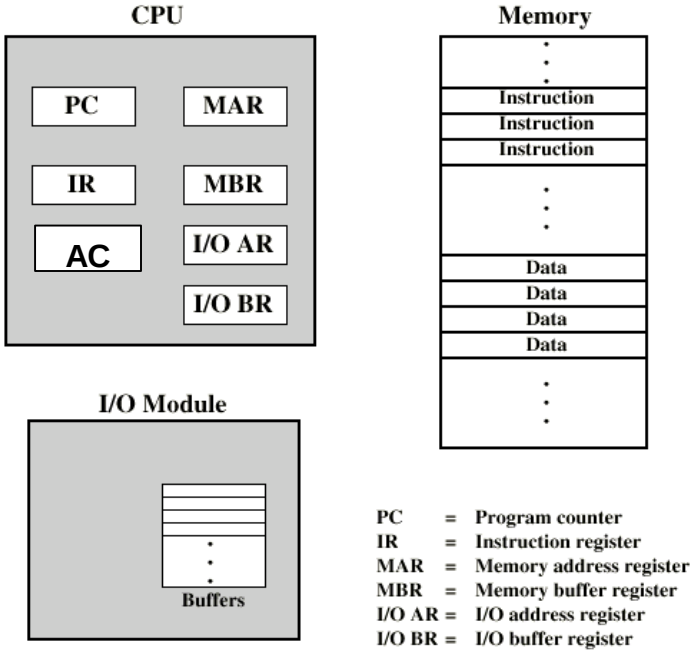


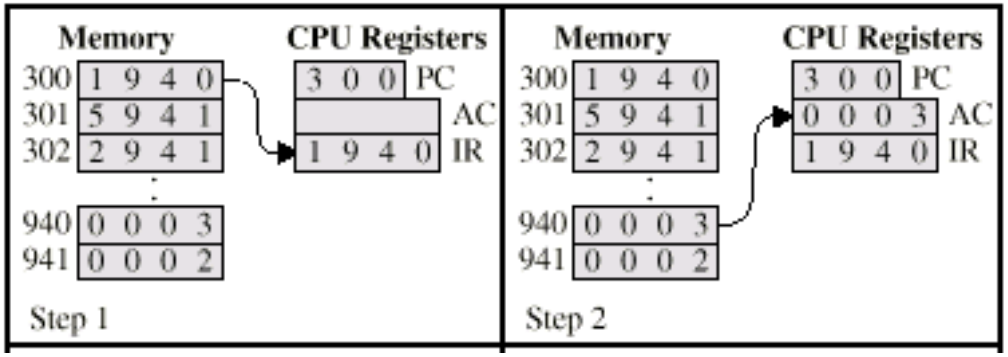
Exemplo de execução de programa



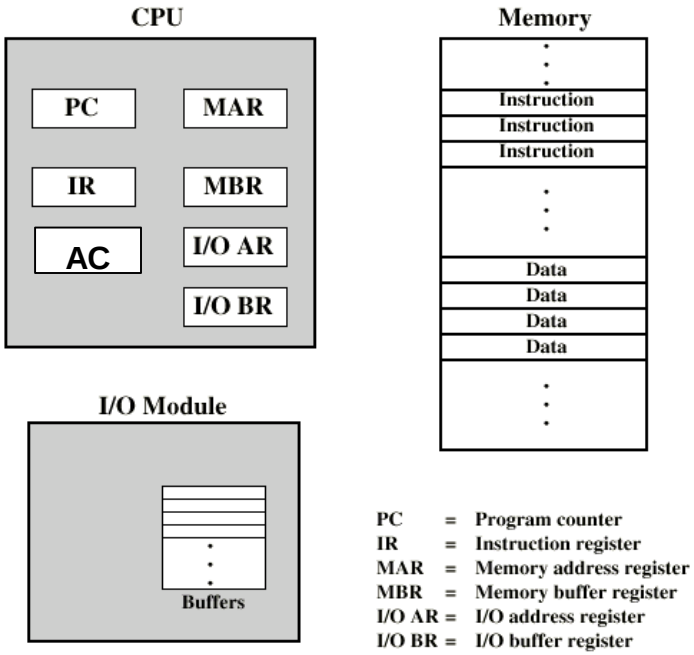


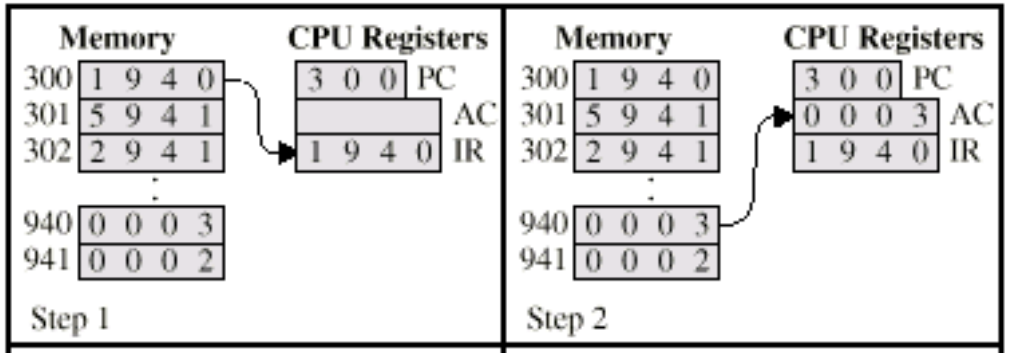
Exemplo de execução de programa





Exemplo de execução de programa



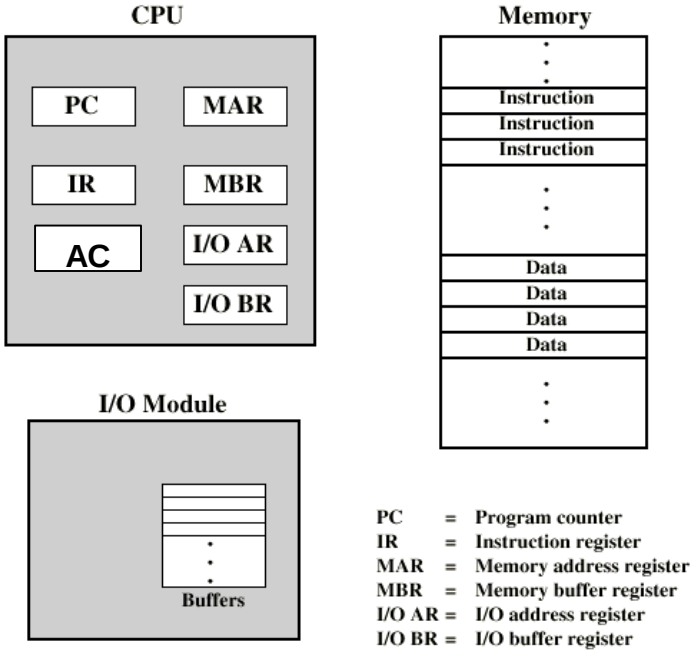


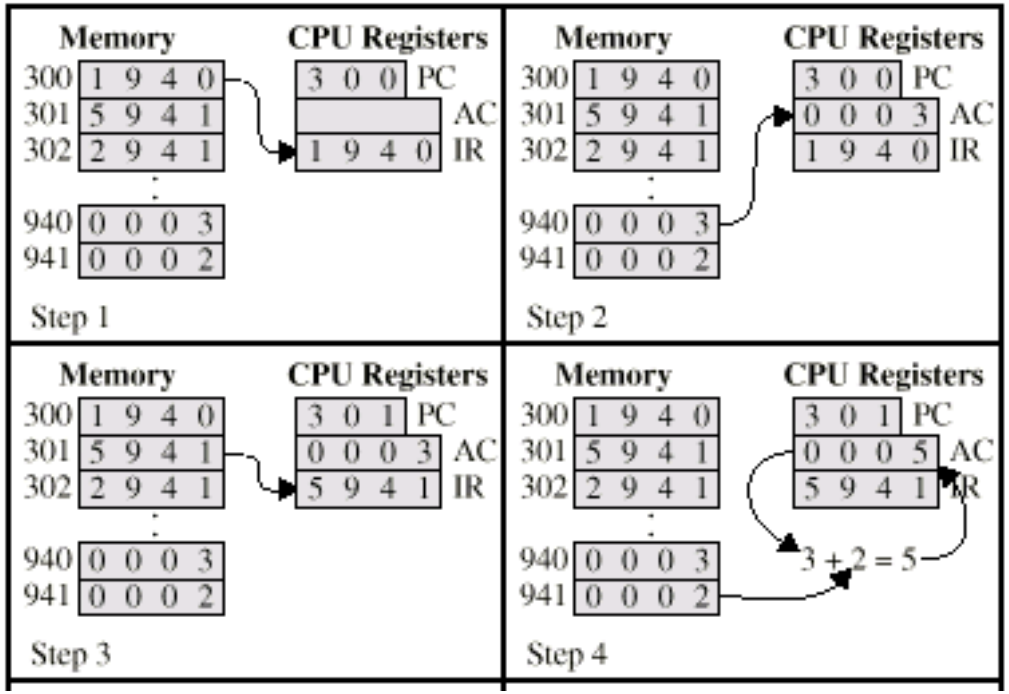
1940 → 1 | 940

OP

Endereço

Exemplo de execução de programa



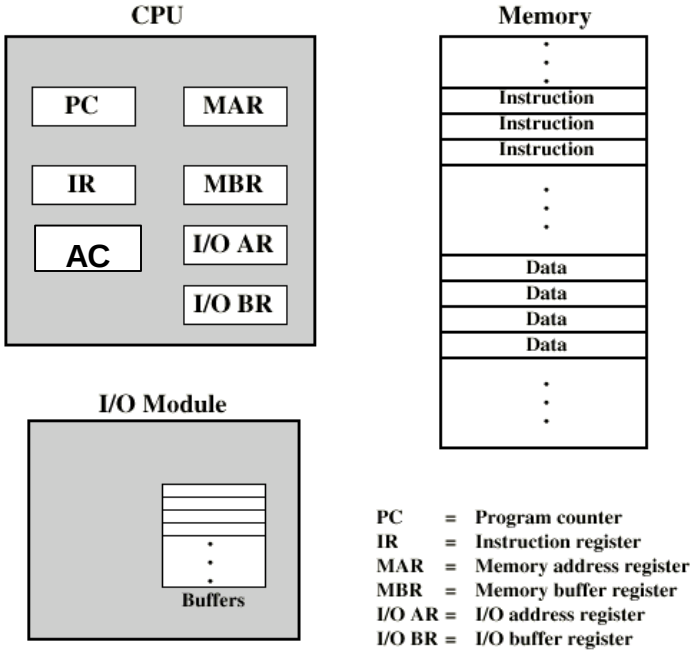


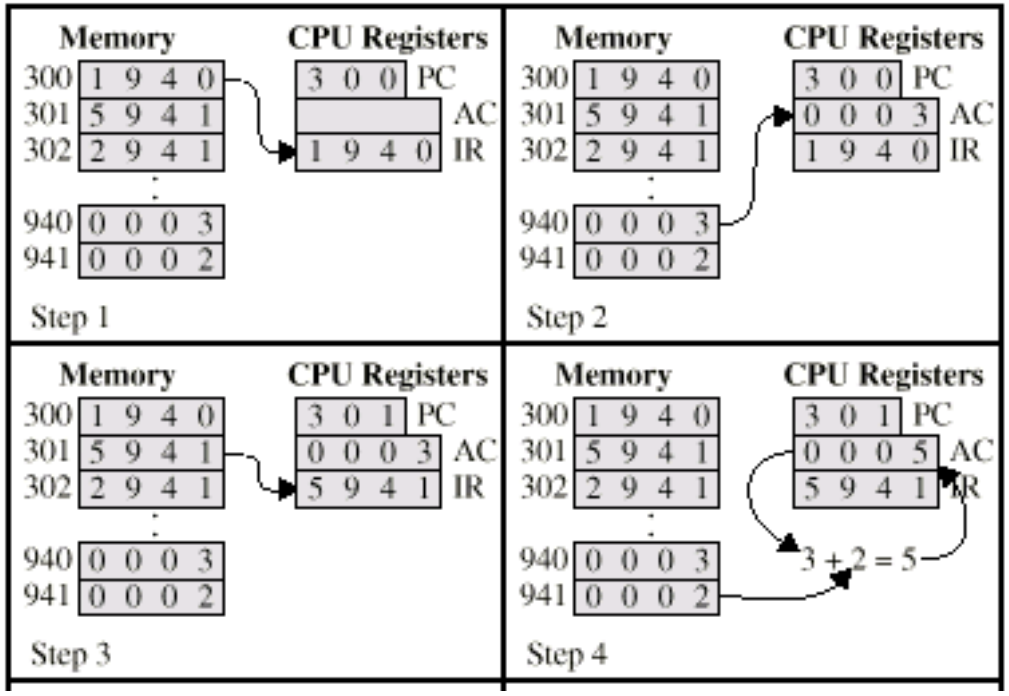
5941 → 5 | 941

OP

Endereço

Exemplo de execução de programa

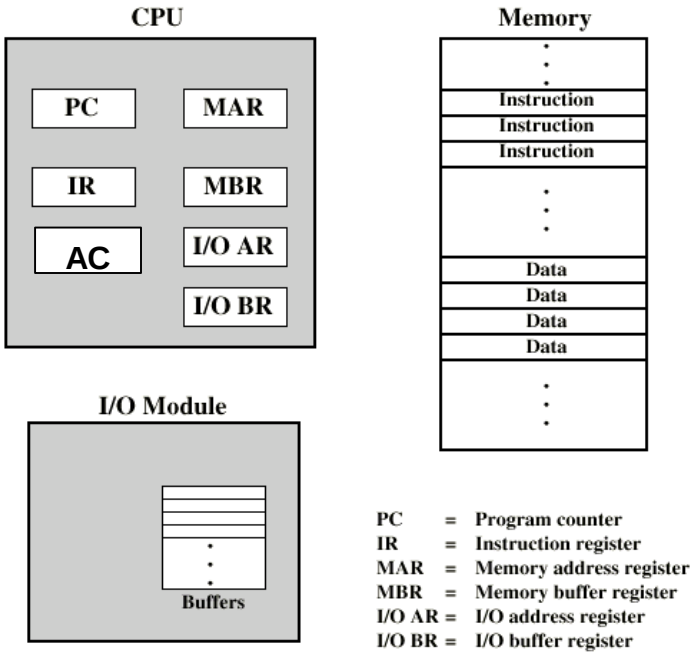


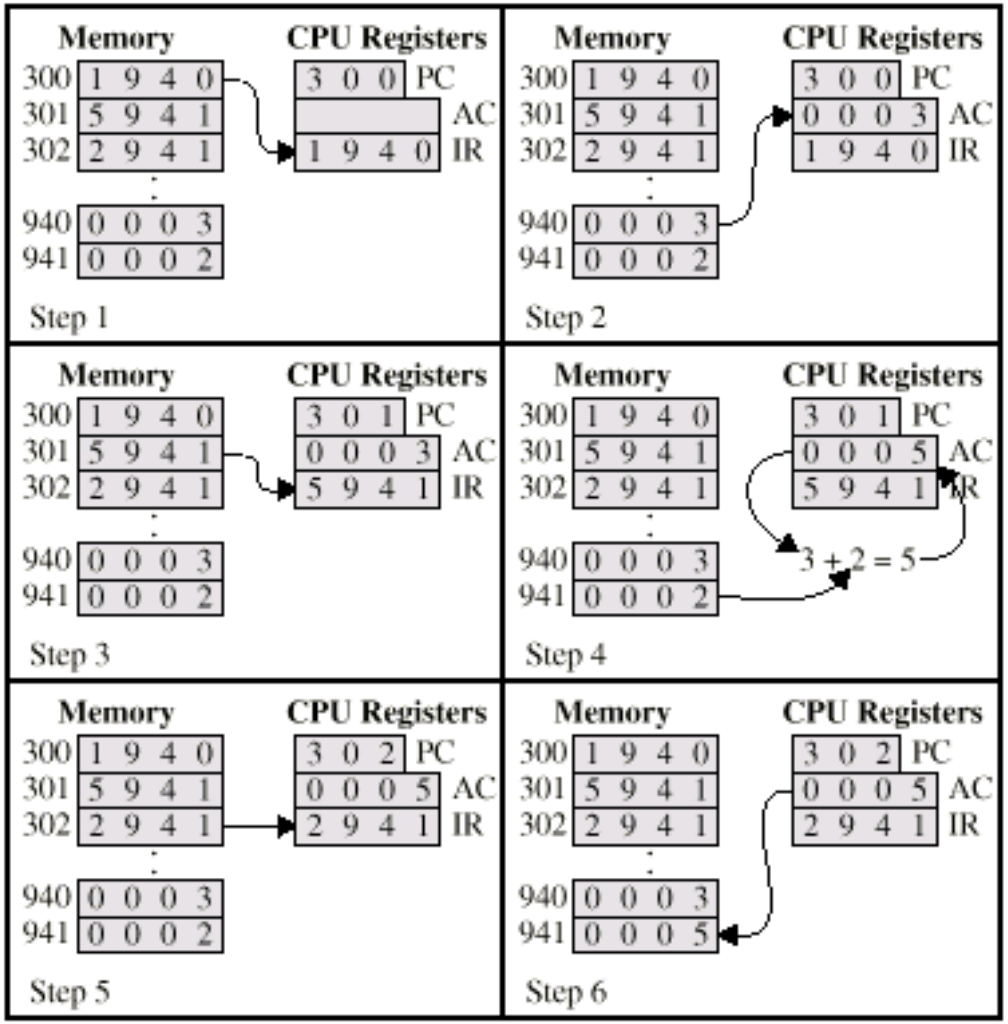


ACC ← ACC + [941]

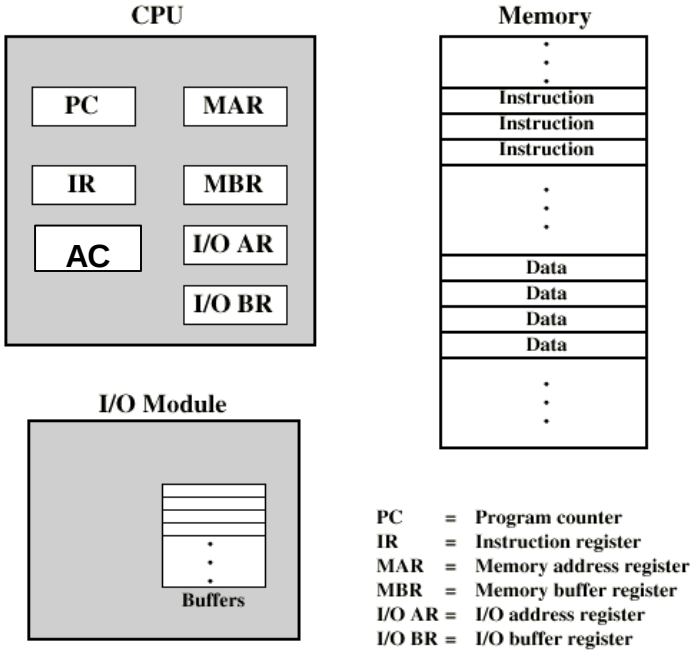
ACC ← 0003 + 0002

Exemplo de execução de programa





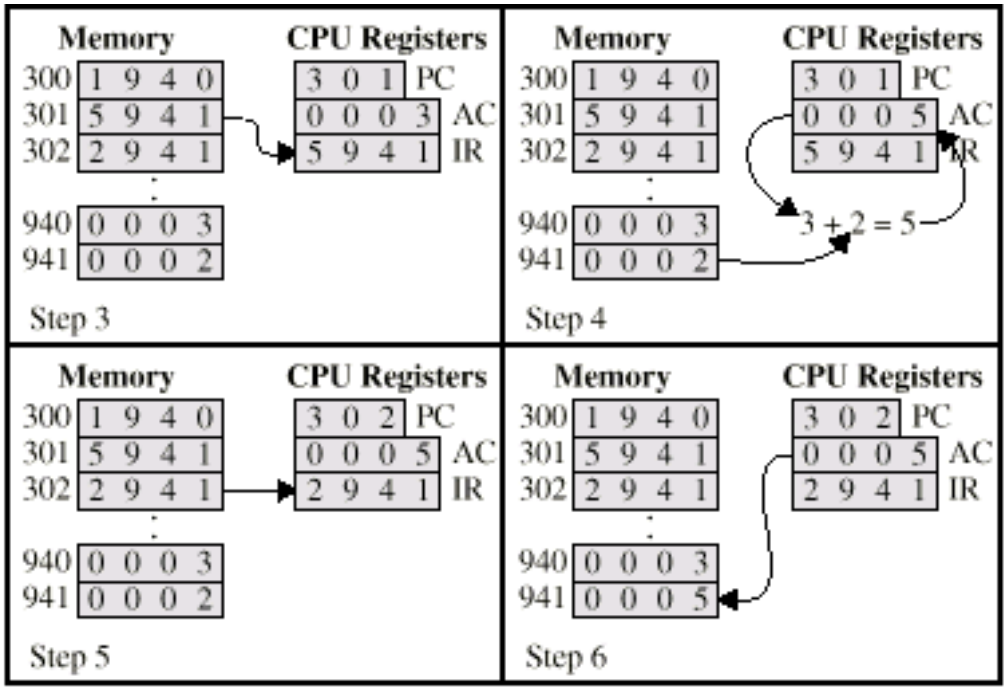
Exemplo de execução de programa



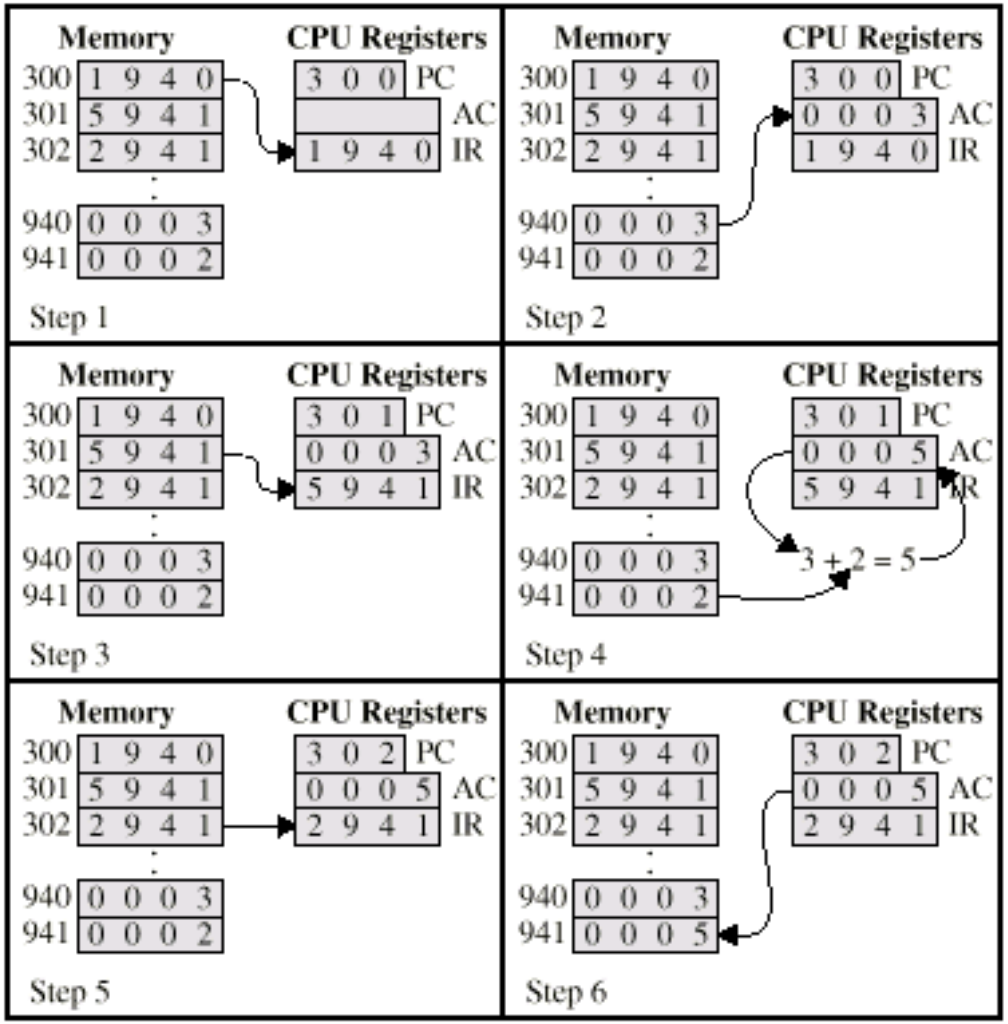
2941 → 2 | 941

OP

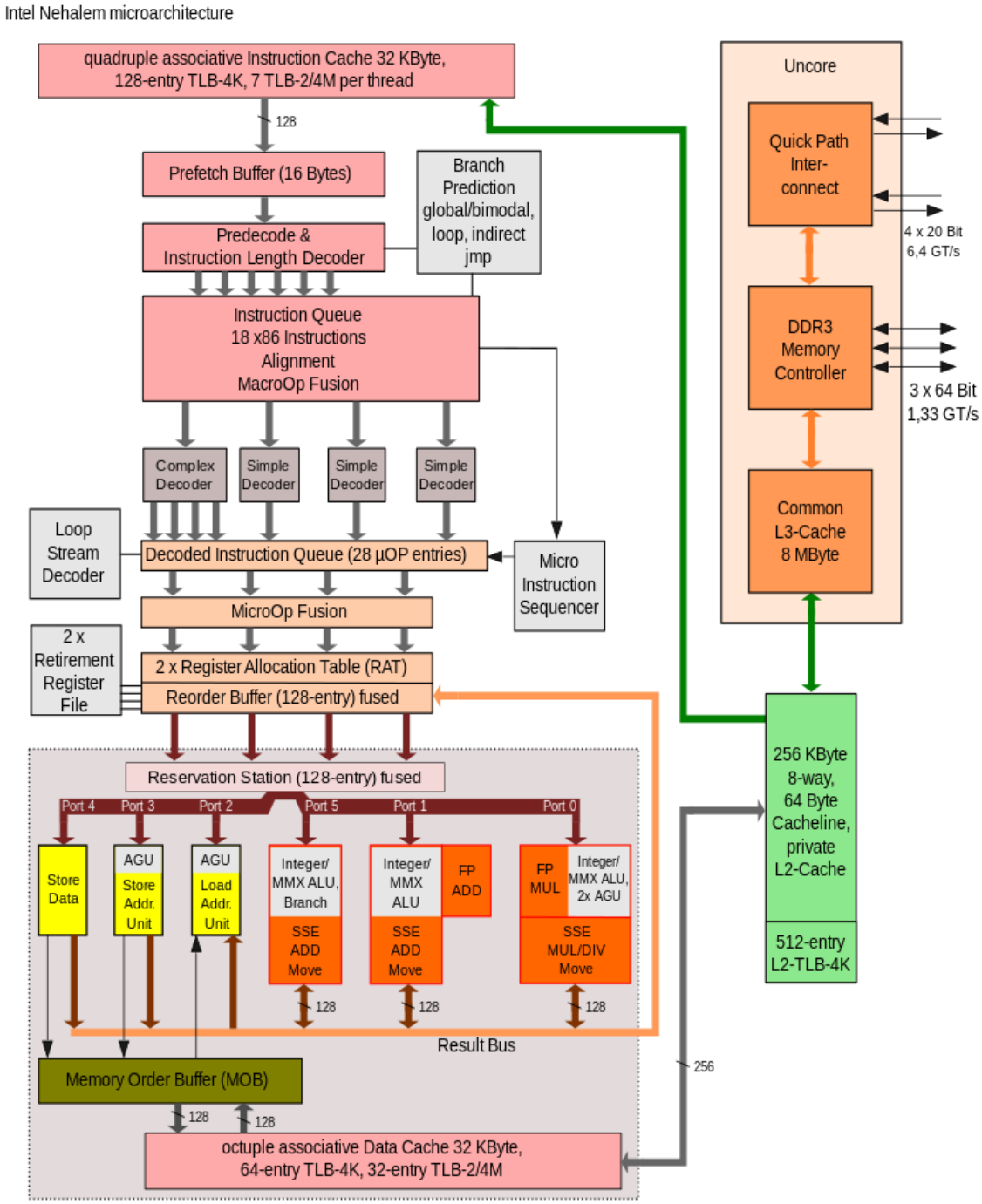
Endereço



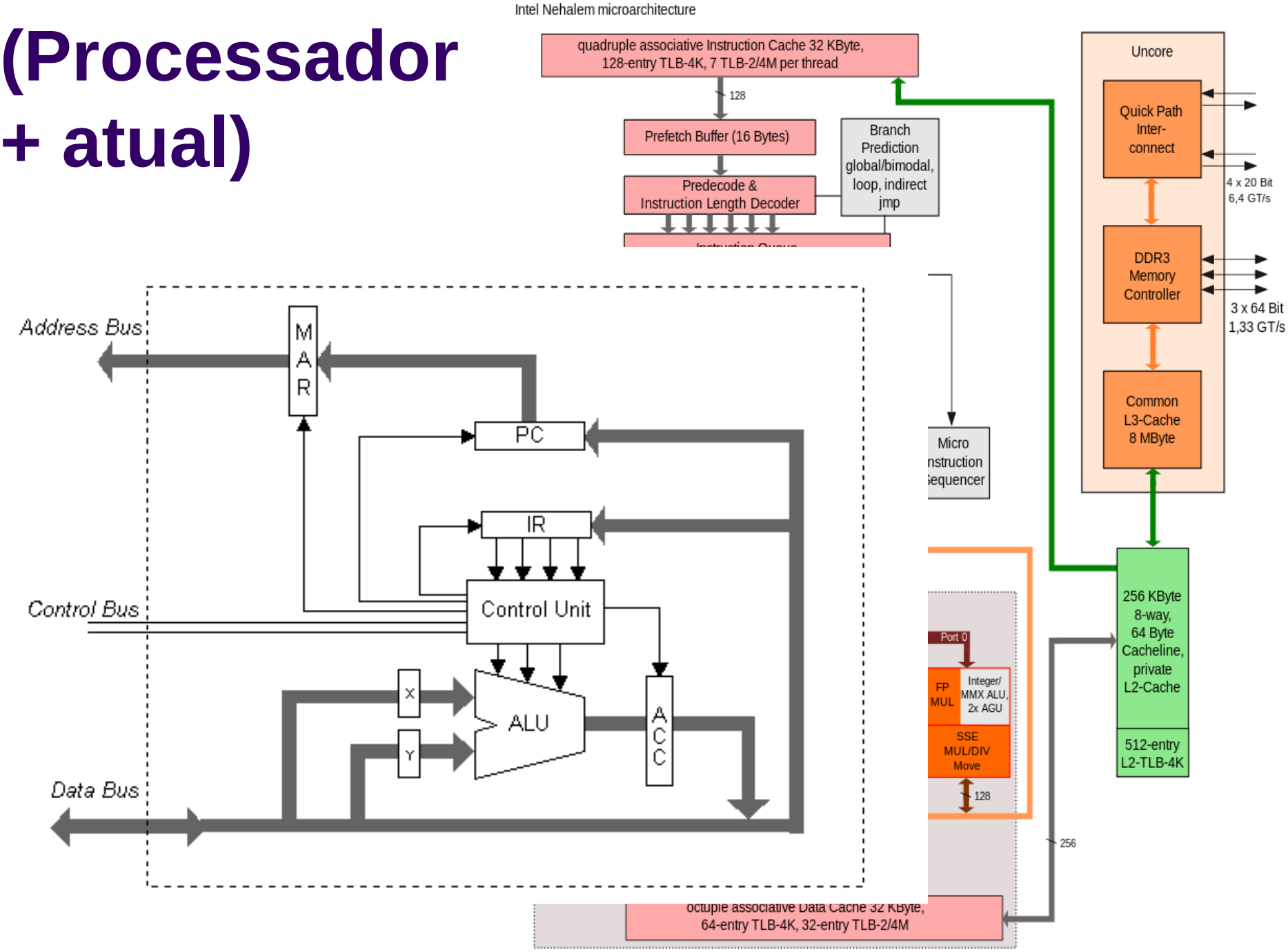
Exercício: descreva, em código “de alto nível”, a operação ilustrada nos 6 passos



(Processador + atual)



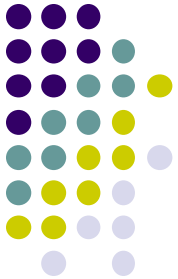
(Processador + atual)



Interrupções

Interrupções

Definição:

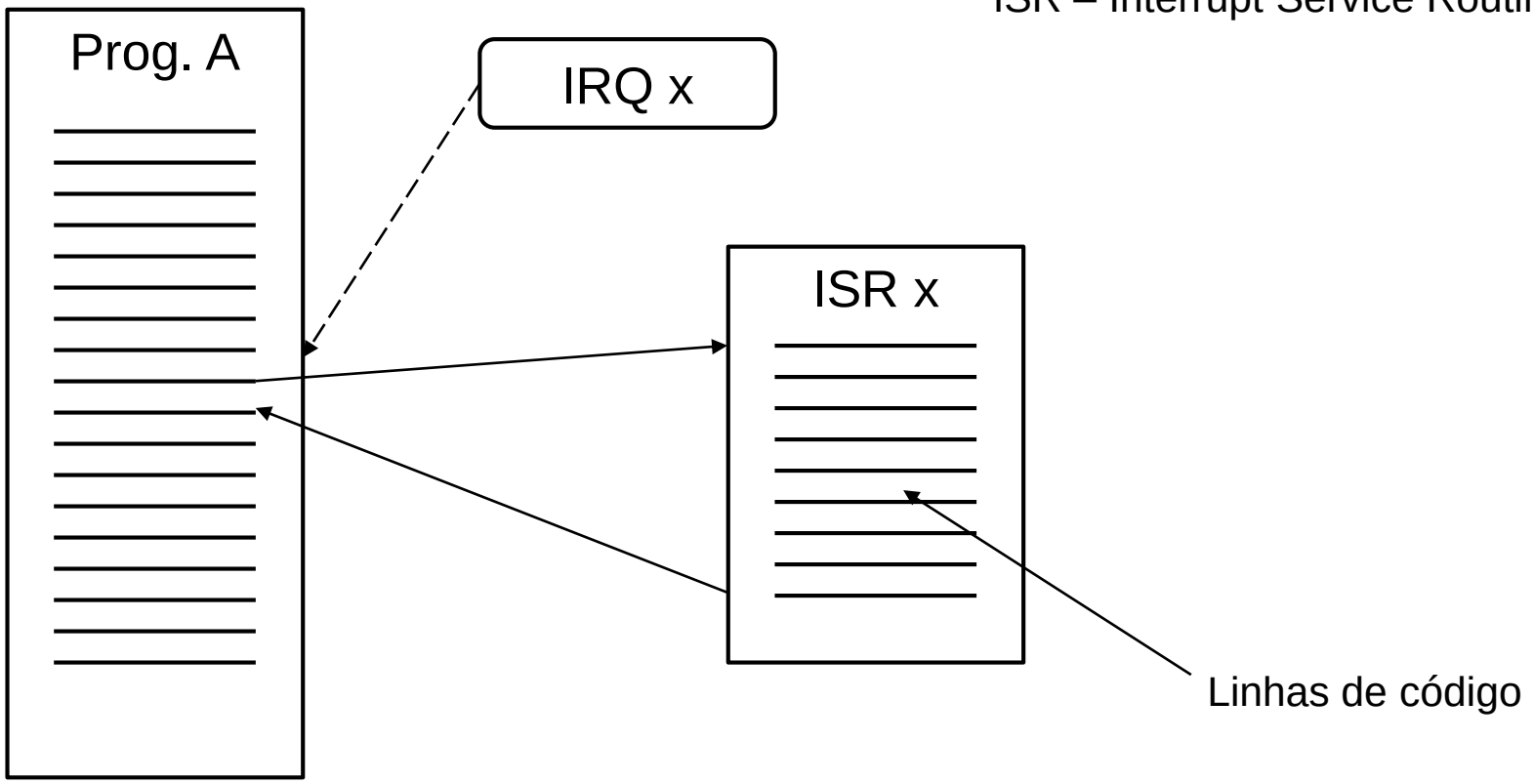


Mecanismo por meio do qual outros módulos (por exemplo: E/S) ou o próprio SW podem **interromper** o processamento normal da CPU.

Interrupções

Definição:

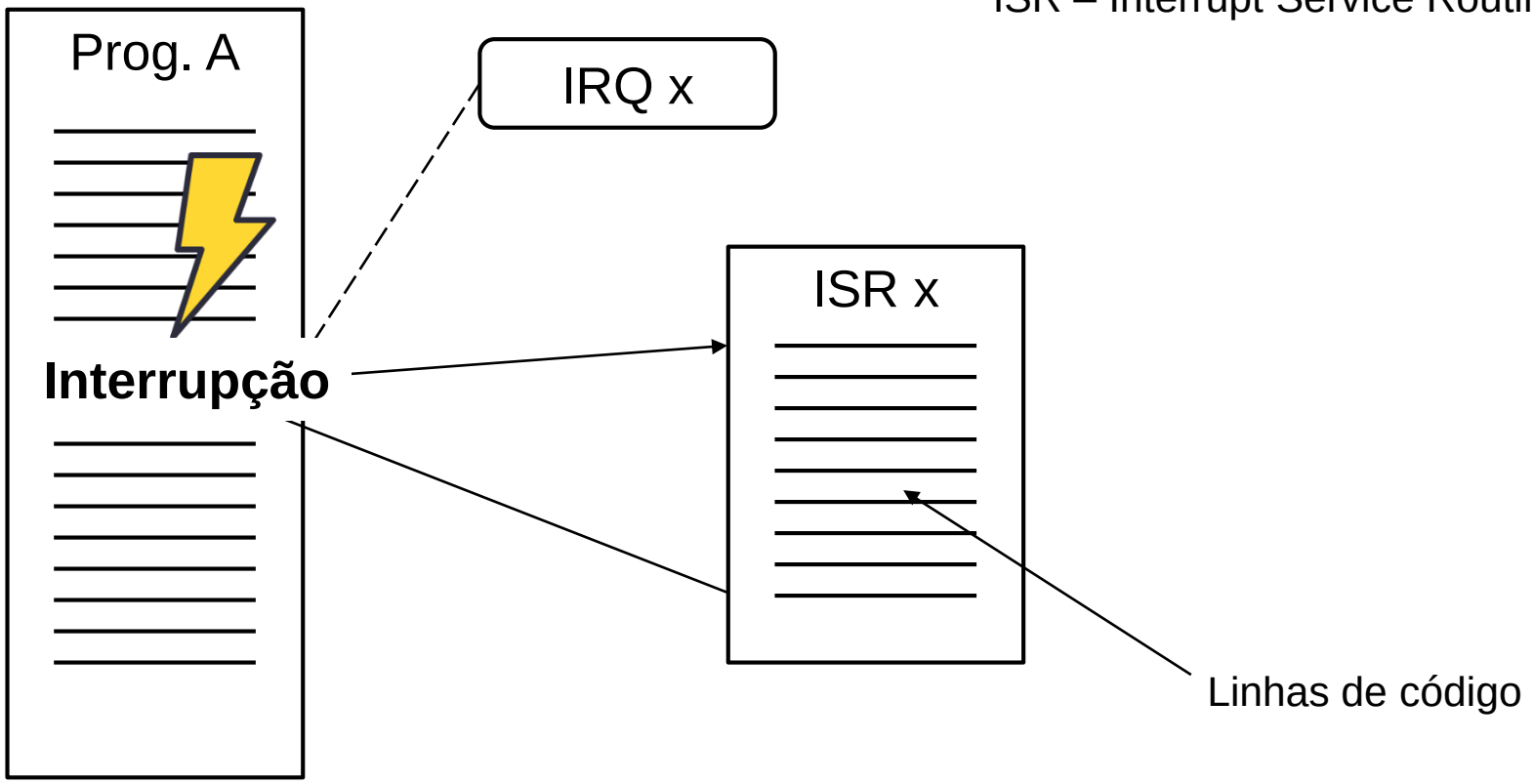
IRQ – Interrupt Request
ISR – Interrupt Service Routine



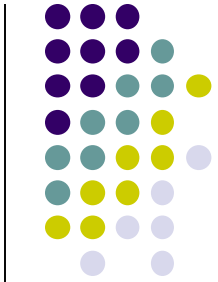
Interrupções

Definição:

IRQ – Interrupt Request
ISR – Interrupt Service Routine



Interrupções

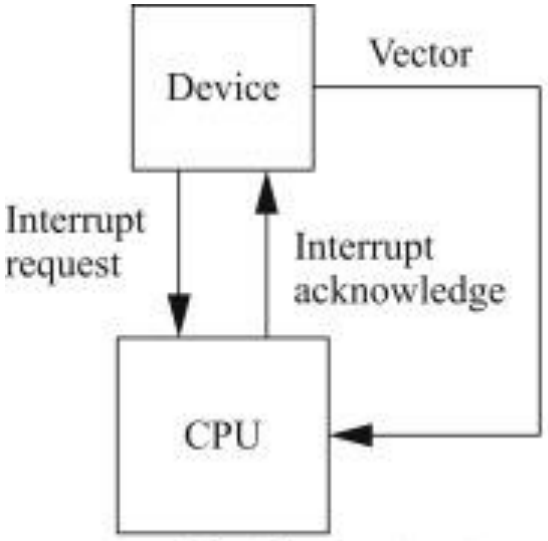


Várias fontes possíveis:

- Interrupção de software
  - Ex.: *overflow*, divisão por zero, acesso proibido à memória
- Interrupção de relógio
  - Gerada pelo relógio interno do processador (System Timer)
  - Permite que o sistema operacional execute funções de multitarefa
- Interrupção de E/S
  - Gerada por um controlador de E/S
- Interrupção de falha de hardware
  - Ex.: erro de paridade de memória

Interrupções

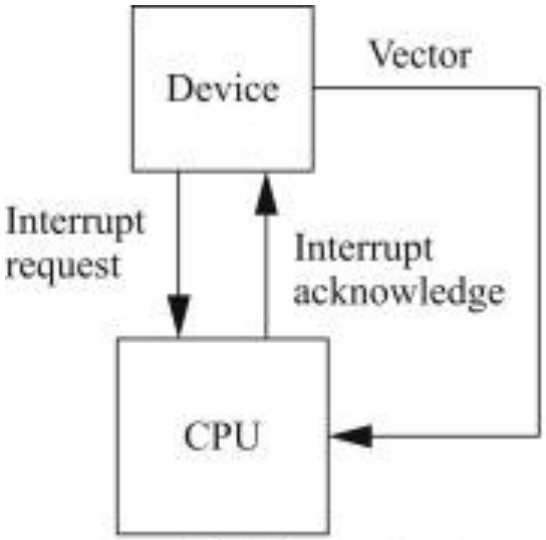
Mecanismo de HW



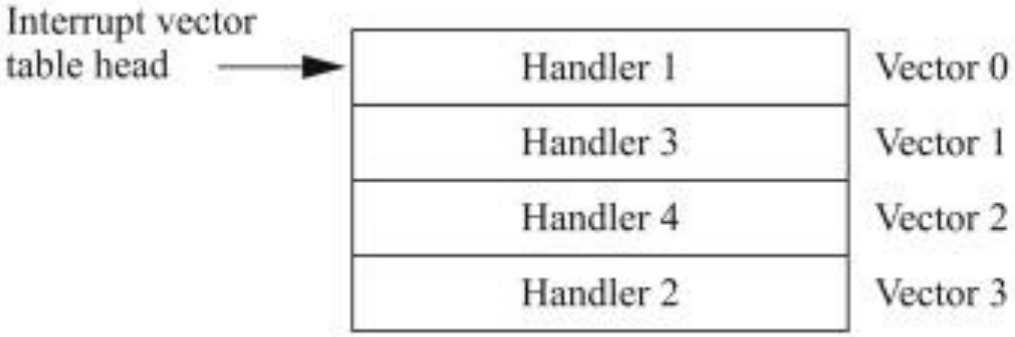
Hardware structure

Interrupções

Mecanismo de HW

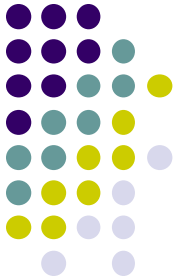


Hardware structure



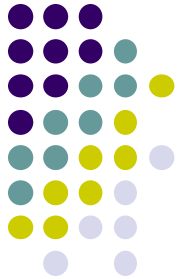
Interrupt vector table

Ex.: x86



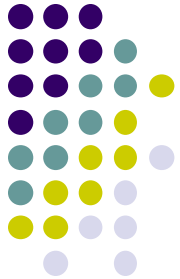
IRQ	Usage
0	system timer (cannot be changed)
1	keyboard controller (cannot be changed)
2	cascaded signals from IRQs 8–15
3	second RS-232 serial port (COM2: in Windows)
4	first RS-232 serial port (COM1: in Windows)
5	parallel port 2 and 3 or sound card
6	floppy disk controller
7	first parallel port
8	real-time clock
9	open interrupt
10	open interrupt
11	open interrupt
12	PS/2 mouse
13	math coprocessor
14	primary ATA channel
15	secondary ATA channel

Ciclo de interrupção



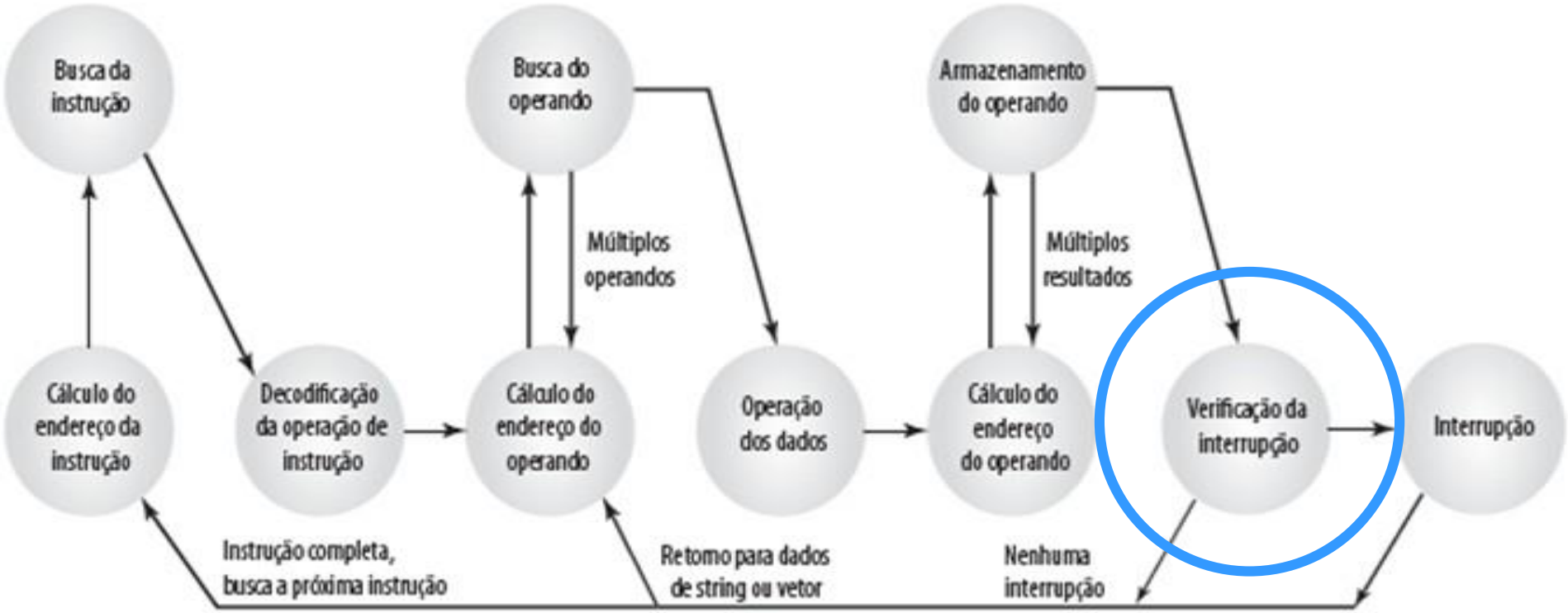
- Acrescentado ao ciclo de instrução
- O processador verifica se ocorreu uma interrupção
  - Indicada por um sinal de interrupção
- Se não ocorreu nenhuma interrupção, busca a próxima instrução
- **Caso exista uma interrupção pendente:**
  - A execução do programa corrente é suspensa;
  - O contexto é salvo;
  - Armazena-se no PC o endereço inicial da rotina de tratamento de interrupção;
  - A interrupção é processada;
  - O contexto é restaurado e o programa interrompido continua.

Ciclo de interrupção

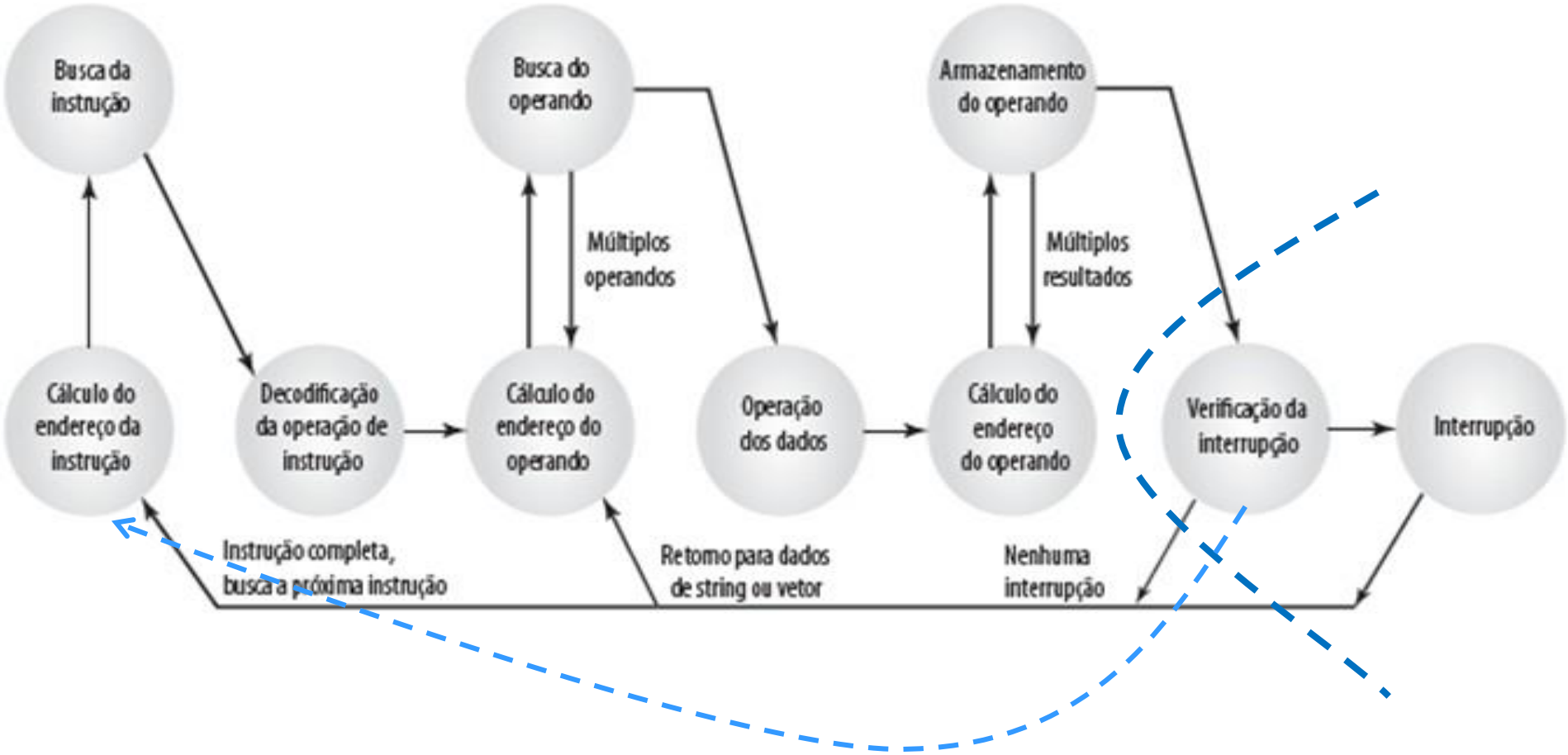


- Acrescentado ao ciclo de instrução
- O processador verifica se ocorreu uma interrupção
  - Indicada por um sinal de interrupção
- Se não ocorreu nenhuma interrupção, busca a próxima instrução
- **Caso exista uma interrupção pendente:**
  - A execução do programa corrente é suspensa;
  - **O contexto é salvo**;
  - Armazena-se no PC o endereço inicial da rotina de tratamento de interrupção;
  - A interrupção é processada;
  - **O contexto é restaurado** e o programa interrompido continua.

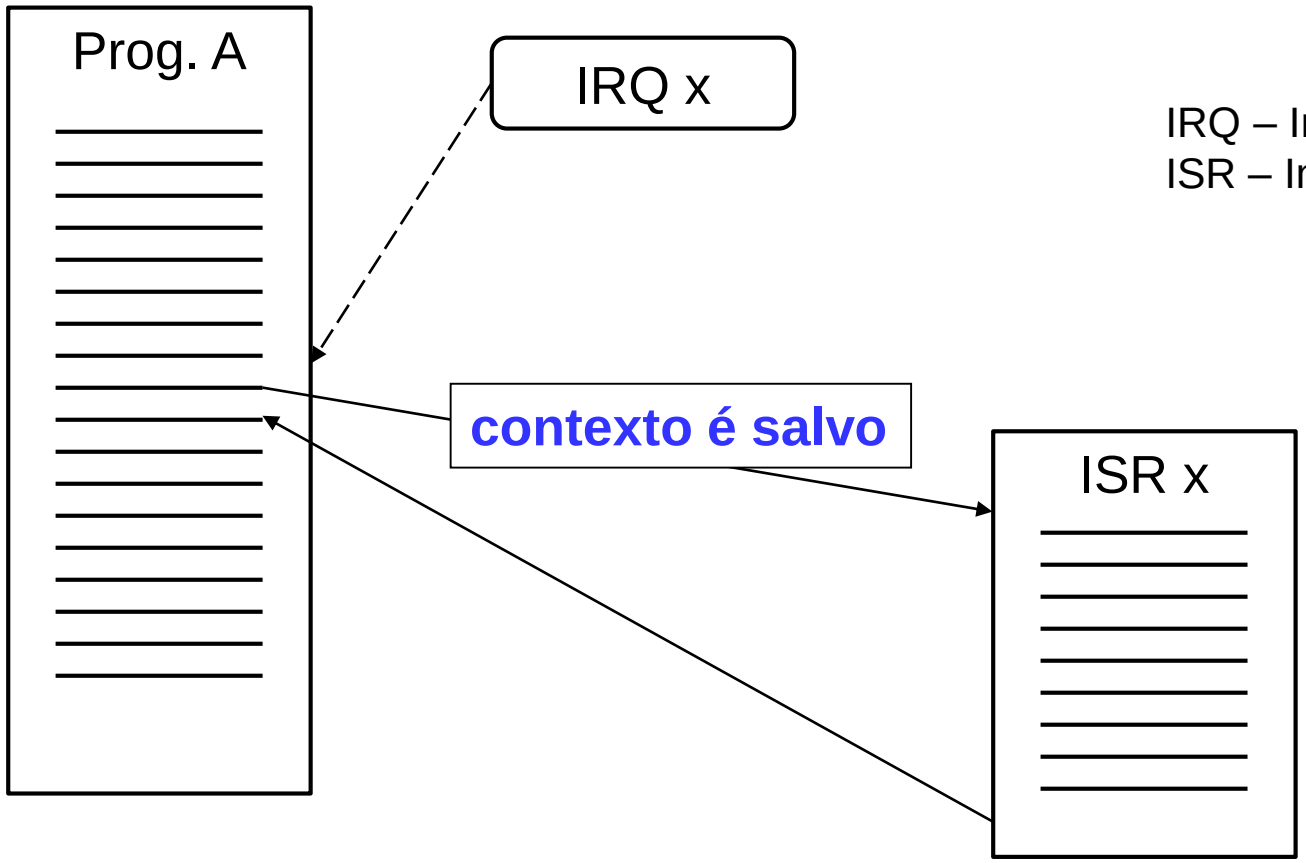
Ciclo de Instrução com Interrupções – Diagrama de Estados



Obs.: Interrupções podem ser DESABILITADAS

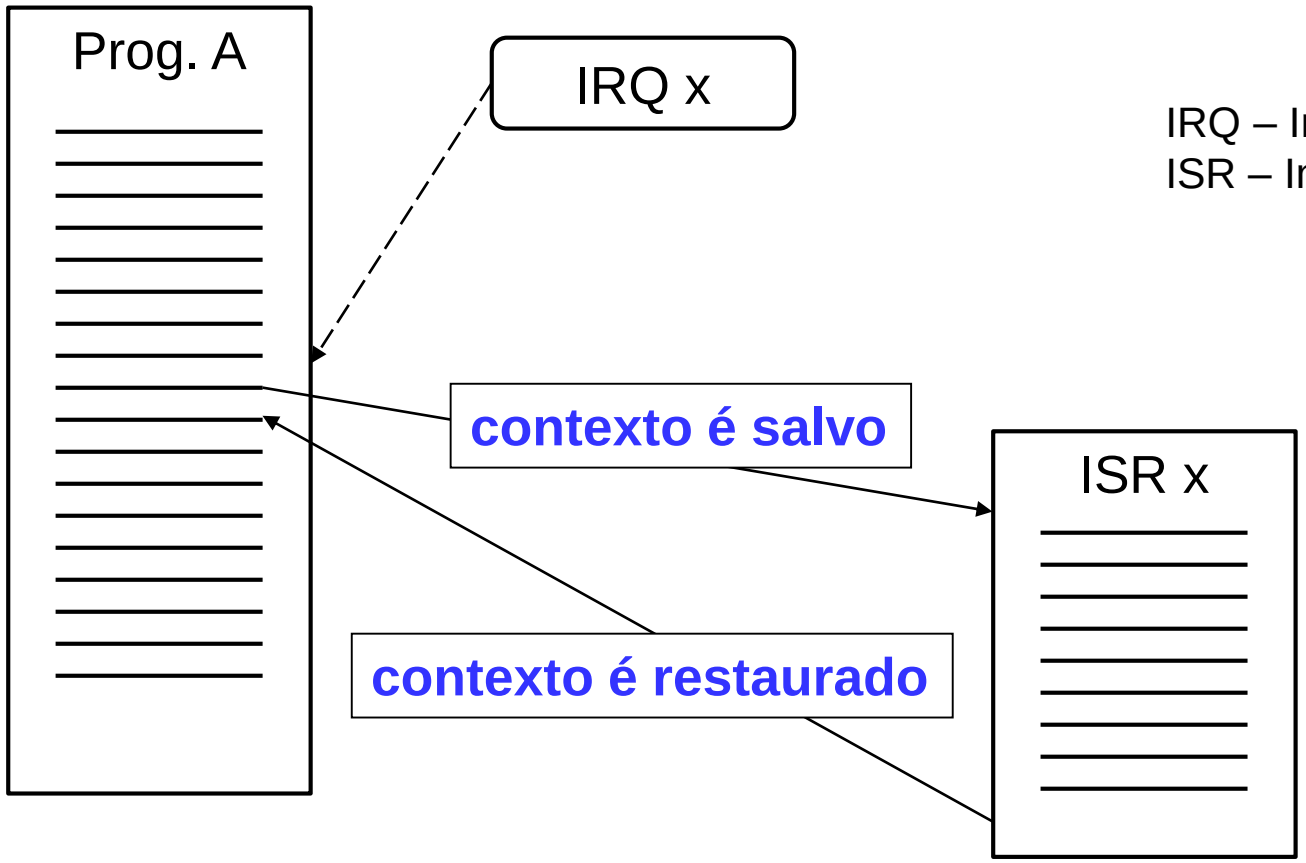


Fluxo de Processamento de uma Interrupção



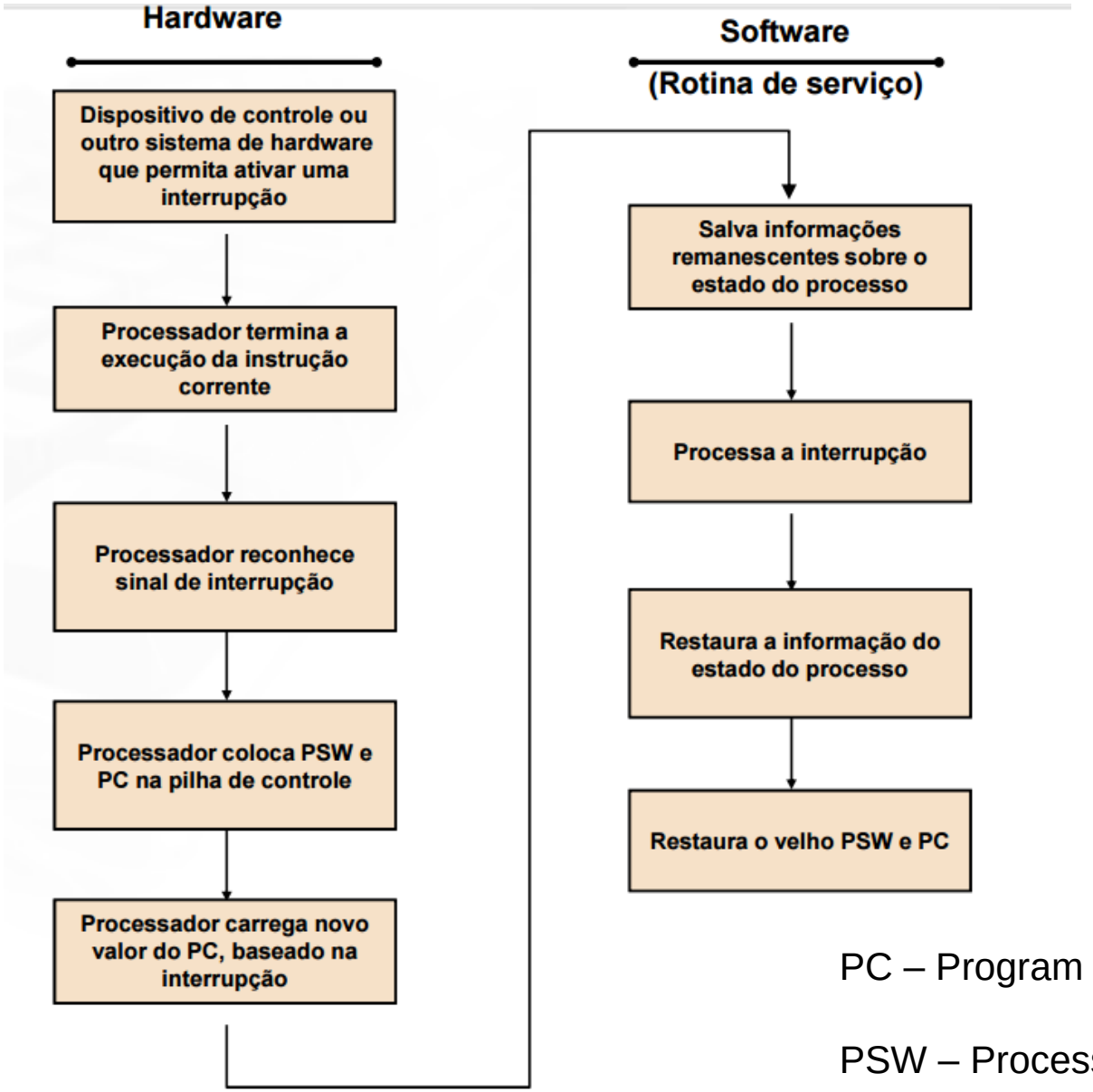
IRQ – Interrupt Request
ISR – Interrupt Service Routine

Fluxo de Processamento de uma Interrupção



IRQ – Interrupt Request
ISR – Interrupt Service Routine

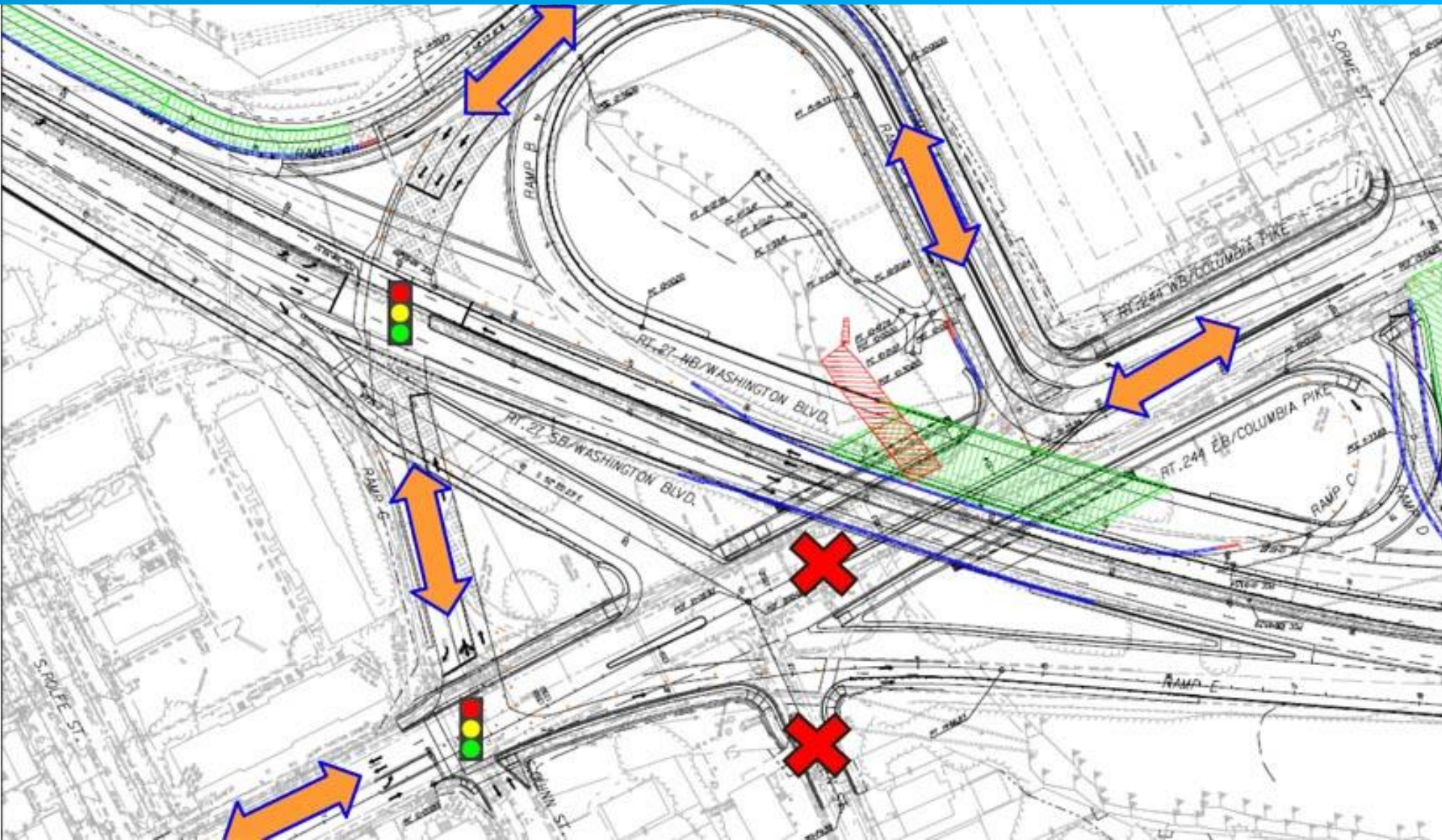
Exemplo de Procedimento



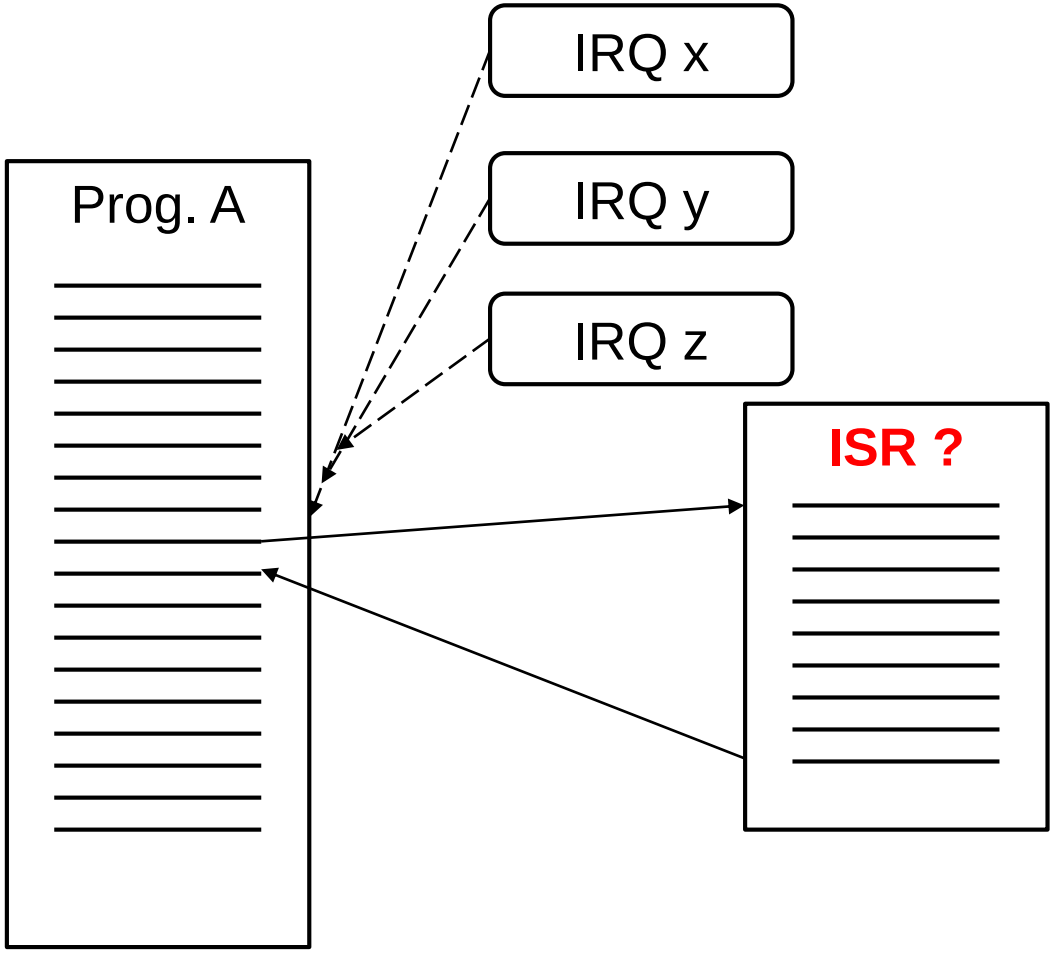
PC – Program Counter

PSW – Processor Status Word

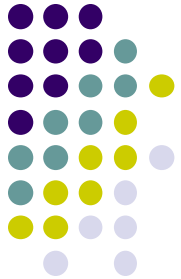
Múltiplas Interrupções



Múltiplas Interrupções

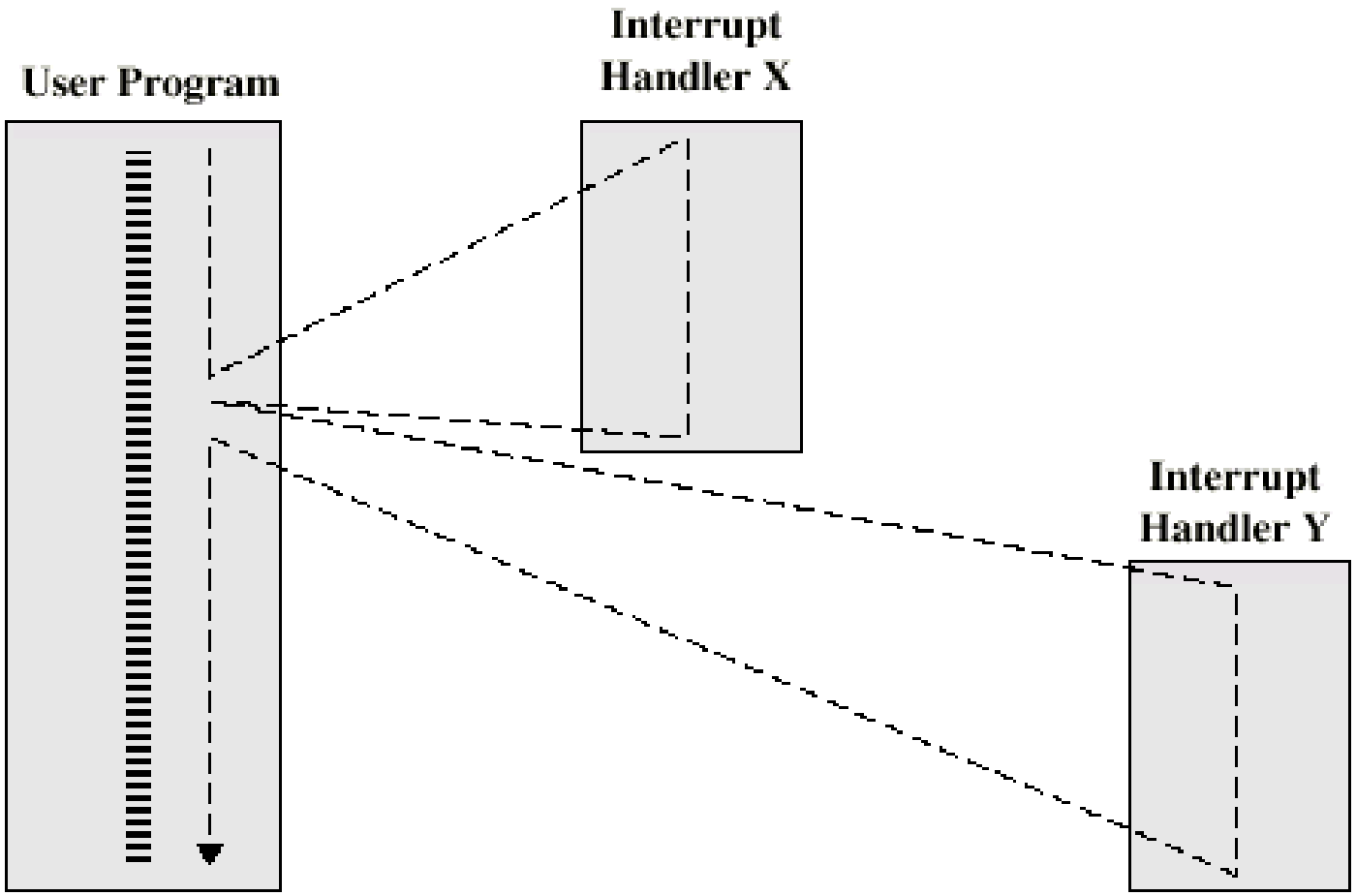


Abordagens para Múltiplas Interrupções

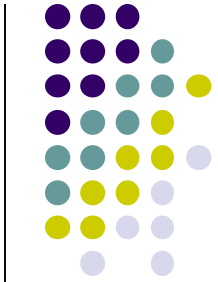


1. Desabilitar interrupções
   - O processador irá ignorar interrupções enquanto uma interrupção é processada
   - As interrupções **continuam pendentes** e são verificadas após o processamento da primeira interrupção
   - As interrupções são tratadas na sequência em que ocorrem
     - Implementação de uma FILA
   - Podem existir NMI (*Non-maskable Interrupt*)
     - Ex.: Reset

Processamento Sequencial de Interrupções



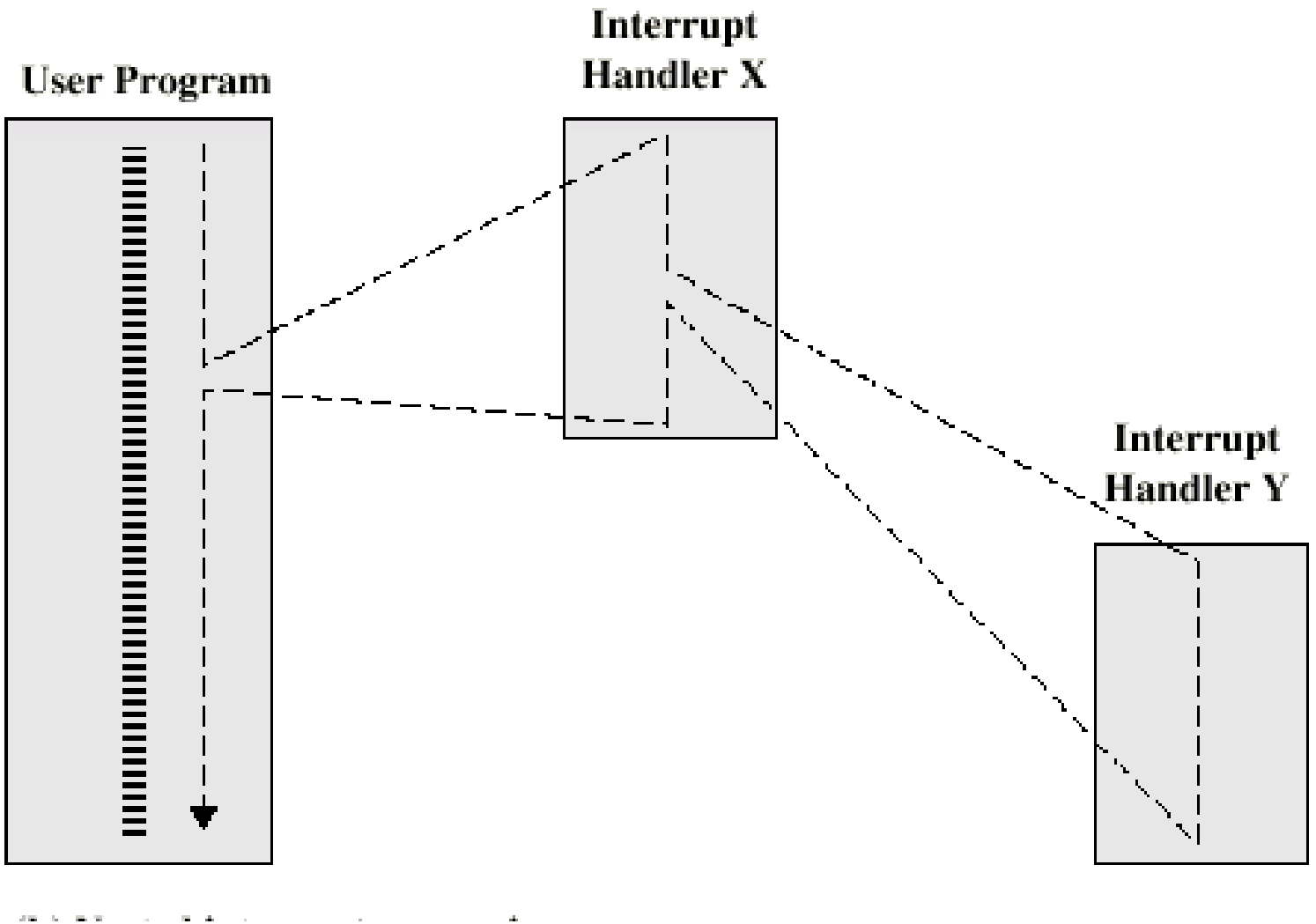
Abordagens para Múltiplas Interrupções



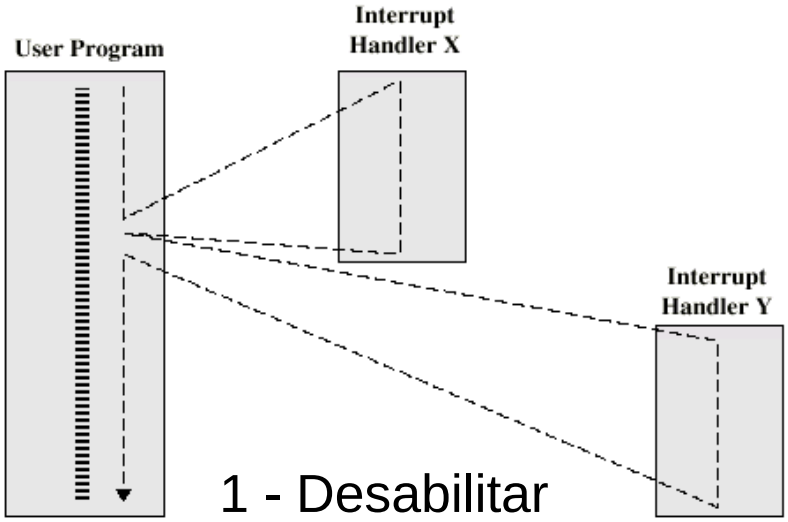
2. Definir prioridades

- Aninhamento de interrupções
- Interrupções de baixa prioridade podem ser interrompidas por outras de prioridade mais alta
- Quando uma interrupção de maior prioridade é processada, o processador retorna à interrupção anterior

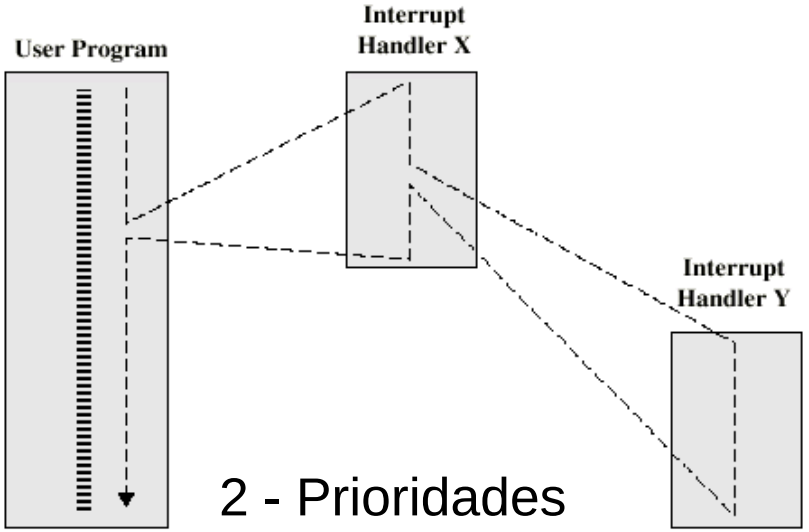
Processamento Aninhado de Interrupções



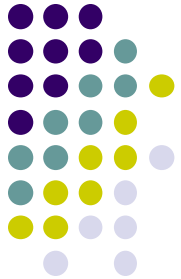
Comparação



1 - Desabilitar

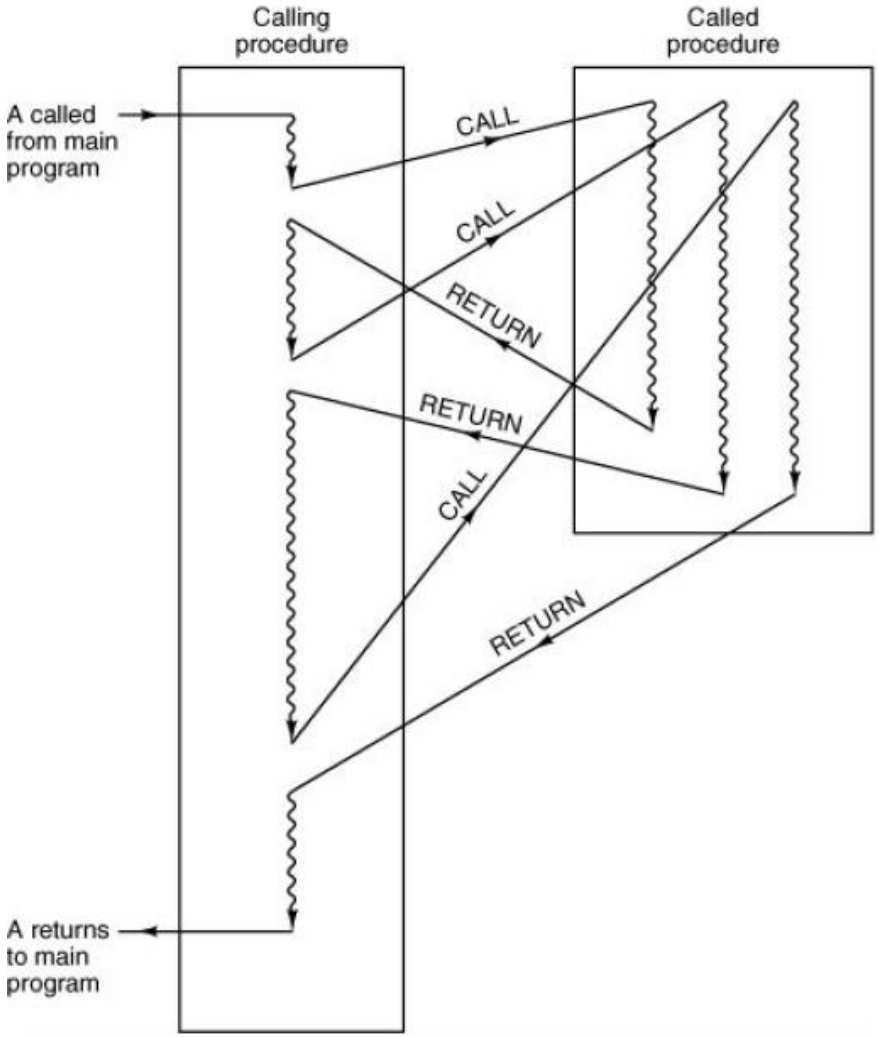


2 - Prioridades

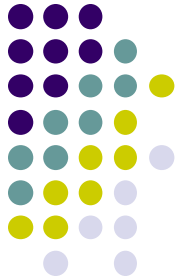


Procedures & Co-rotinas

Procedures

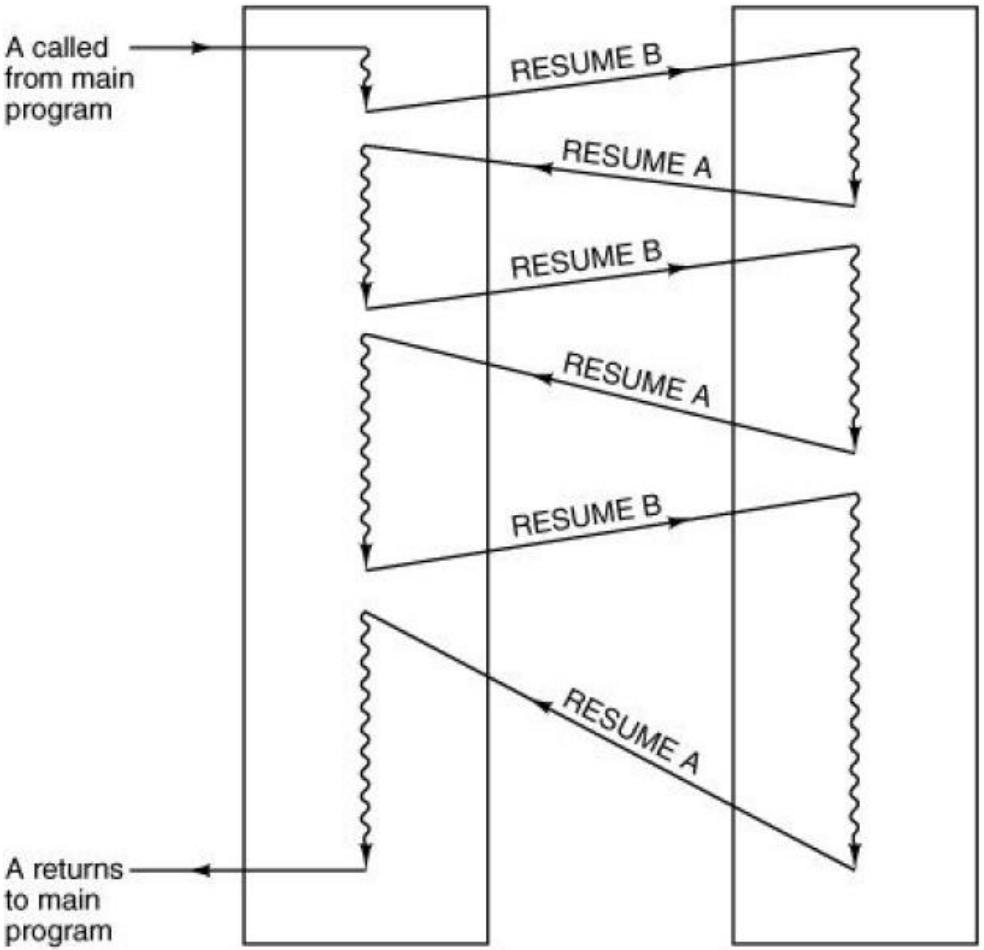

Calling procedure
Called procedure
A called from main program
CALL
CALL
RETURN
RETURN
CALL
RETURN
A returns to main program

Procedures



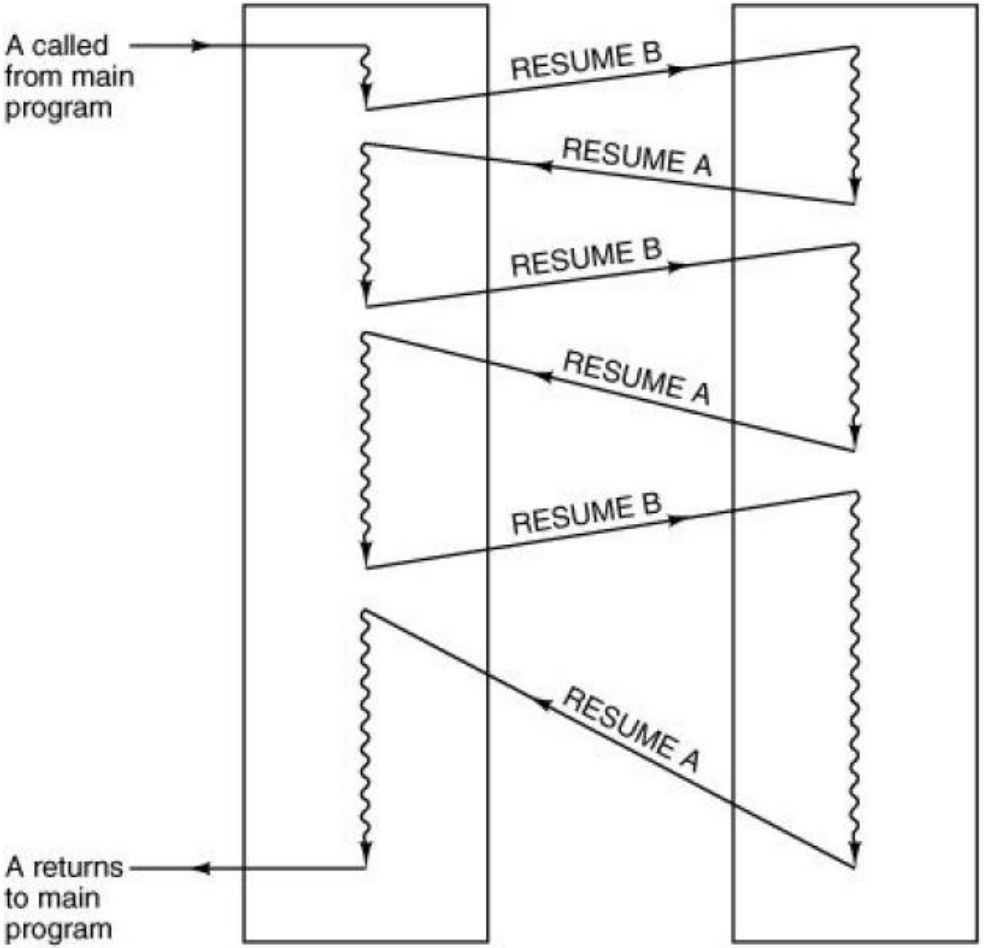
```
static int a = 8;   /*Static Storage*/
extern c;

static int func1 (int x)
{
        int t = 8;/* Local Variable */
        return (x+t);       /* Return */
}

int main()
{
        int b = 0;
        b = func1(a);
        c = b;
        return(0);
}
```

Co-rotinas

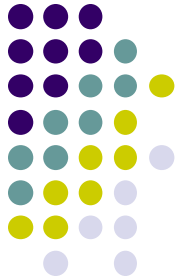

A called
from main
program
RESUME B
RESUME A
RESUME B
RESUME A
RESUME B
RESUME A
A returns
to main
program

Co-rotinas

Cooperação

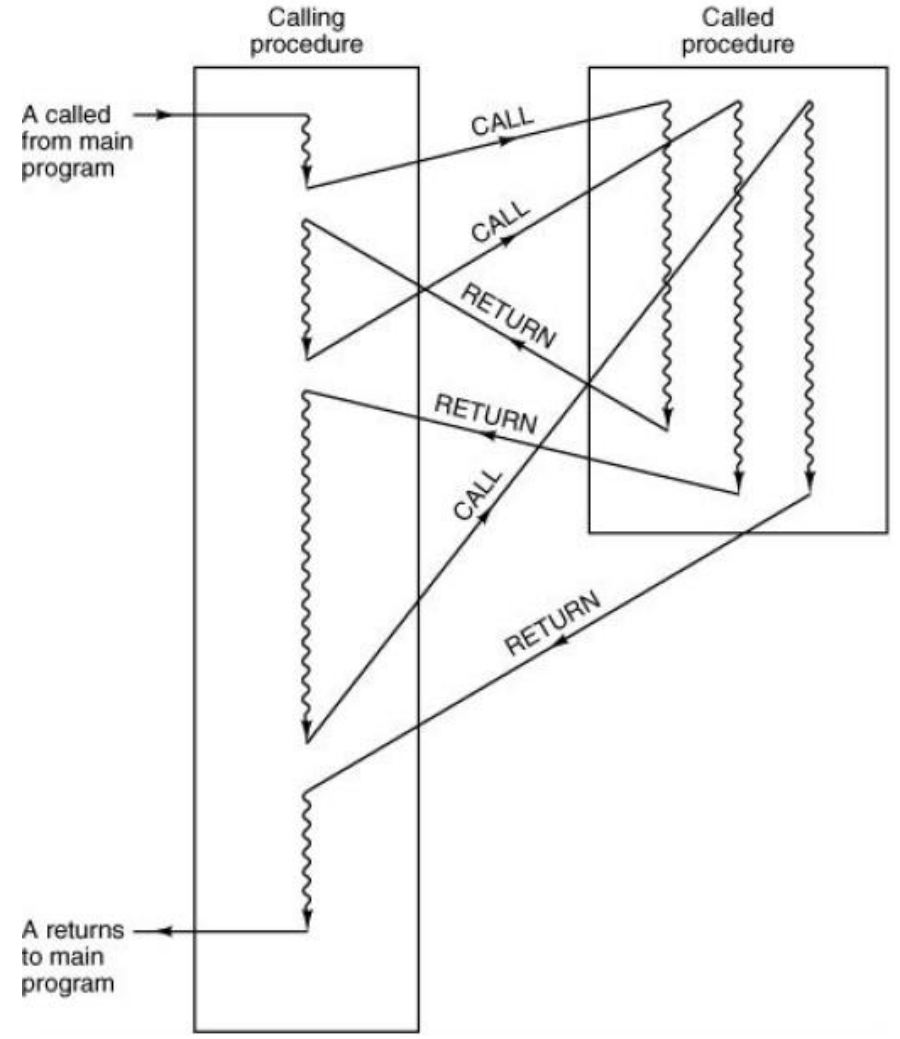


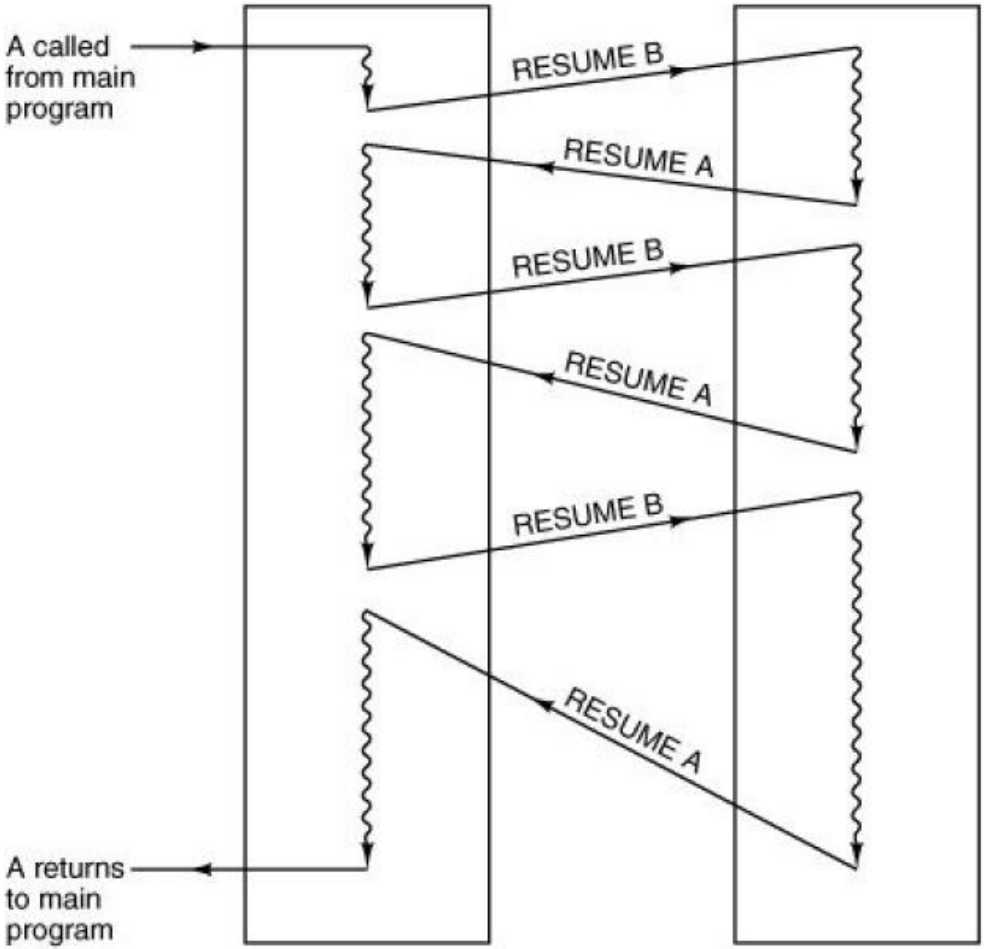
Co-rotinas



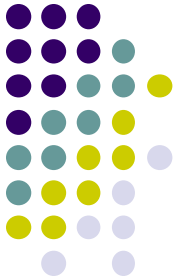
```
var q := FILA

coroutine Produtor
    loop
        while (q != CHEIA)
            Cria Item
            Adiciona Item a q
        yield to Consumidor

coroutine Consumidor
    loop
        while (q != VAZIA)
            remove Item de q
            Consome Item
        yield to Produtor
```

Procedures & Co-rotinas





Perguntas



- O que é um programa?
- Diferencie HW de propósito geral de WH dedicado?
- O que contém um ciclo completo de instrução?
- O que são interrupções?
- Quais as abordagens para as múltiplas interrupções?
- Qual a diferença entre *Procedure* e Co-rotina?