

Interrupção

INF01112

2008

Raul Weber/J.Netto

Tipos de transferência de E/S

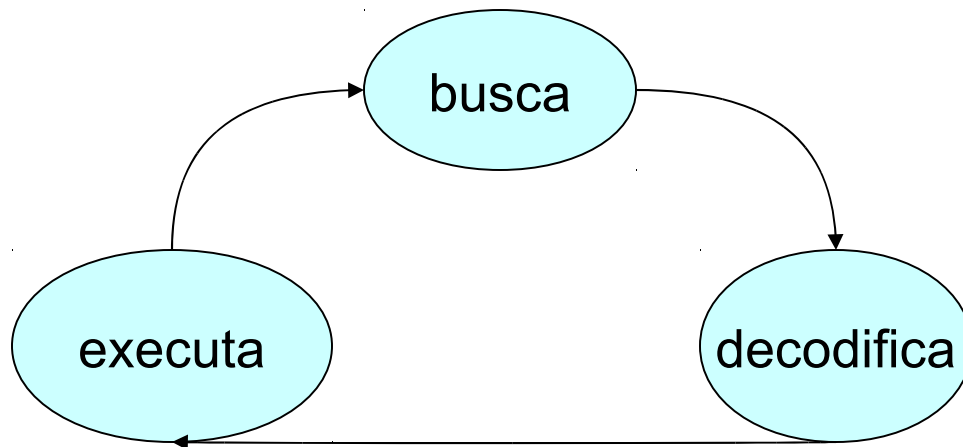
- Controle direto do processador
 - polling
 - interrupção
- Acesso direto a memória: DMA
- Interrupção
 - E/S controlada diretamente pelo processador
 - significa que o **processador** participa da transferência de dados,
 - que o **processador** é informado quando uma transferência direta a memória foi concluída,
 - ou de alguma outra situação excepcional que exija atenção

Origem da Interrupção

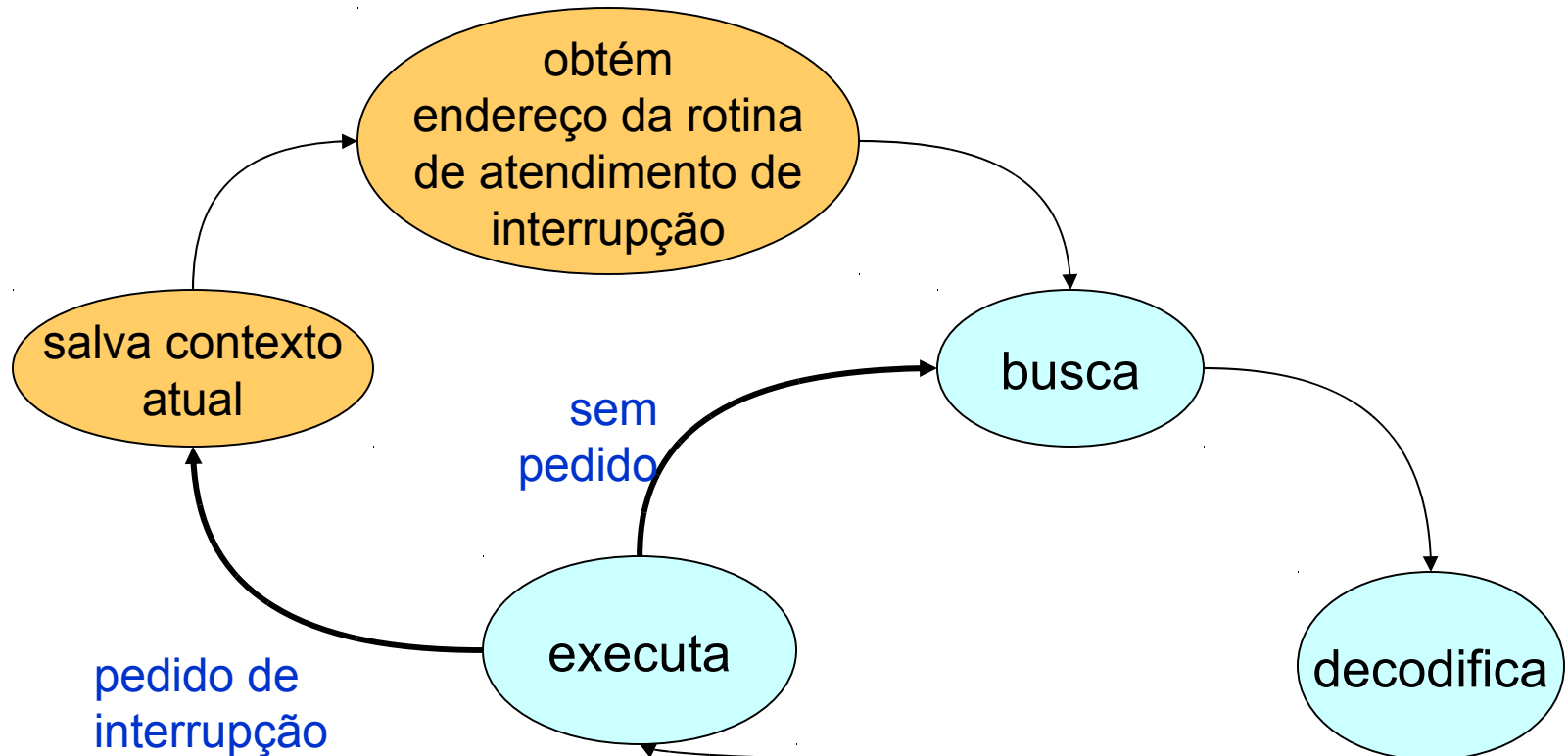
- Microprocessador
 - evento gerado internamente ao processador
 - (divisão por zero, falta de página, etc)
- Entrada e Saída (Hardware)
 - pedido de interrupção
 - assíncrona ao processador
 - externa
- Programada (Software)
 - instrução INT
 - síncrona
 - interna

Ciclo de operação convencional

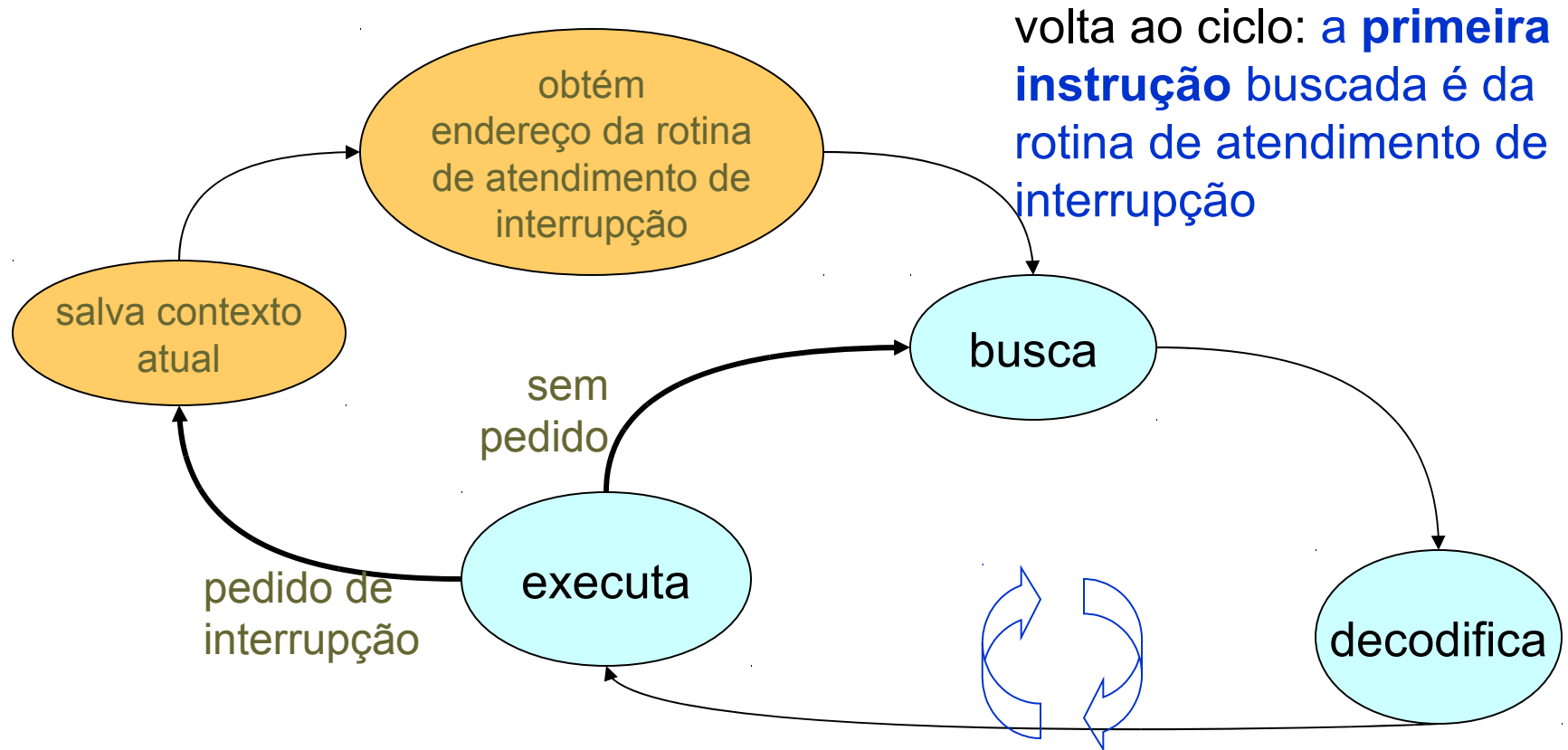
onde o ciclo pode ser interrompido mantendo o estado do processador consistente?



Ciclo de operação com interrupção



Ciclo de operação com interrupção



permanece no ciclo até executar retorno de interrupção

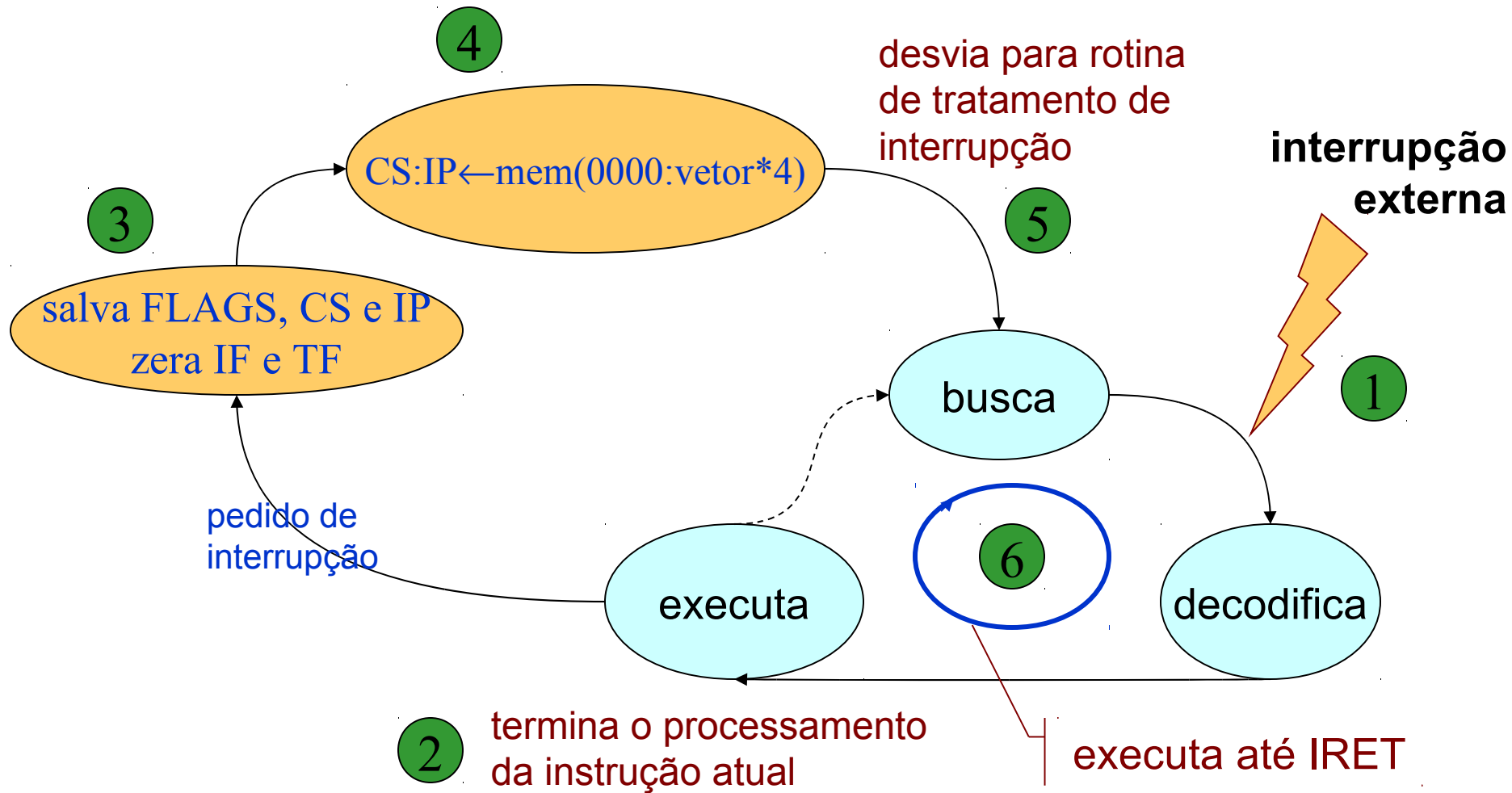
Atendimento de Interrupção exemplo Intel x86

- Se for interrupção externa
 - termina o processamento da instrução atual (se for MOV ou POP para reg. de segmento, executa também a próxima instrução)
 - salva na pilha o registrador de FLAGS
 - zera os flags IF e TF
 - salva na pilha os reg. CS e IP (endereço de retorno)
 - obtém os novos valores de CS e IP a partir da tabela de vetores de interrupção:

$CS:IP \leftarrow \text{mem}(0000:\text{vetor} * 4)$

- desvia para rotina de tratamento de interrupção
 - retorno através da instrução IRET

Tratamento de Interrupção



Rotina de tratamento de interrupção

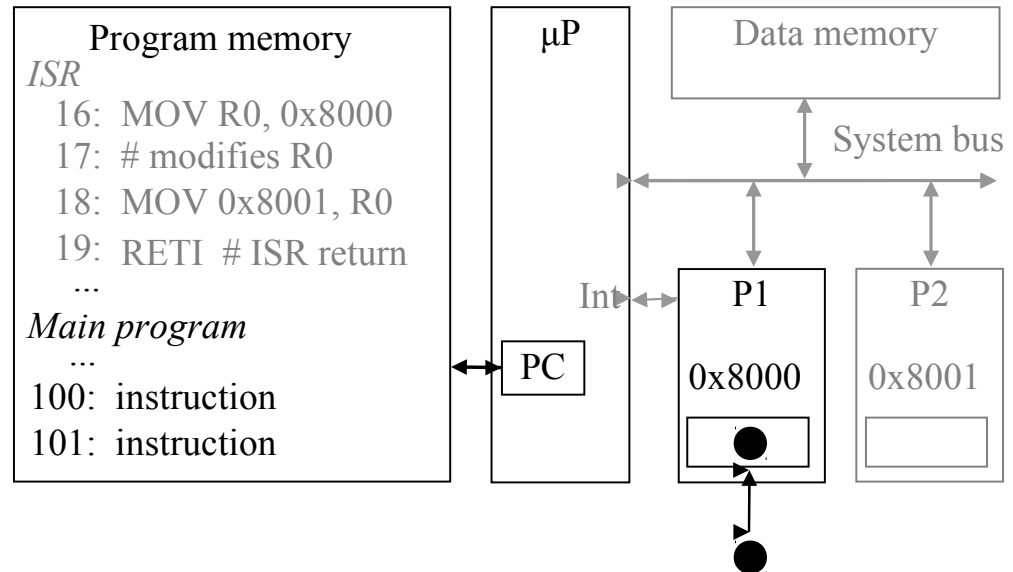
- rotina do usuário, do sistema operacional ou do drive do dispositivo
- características
 - deve ser pequena
 - deve salvar os registradores do processador (**antes de usá-los**)
 - determina se **libera** o sistema de interrupção ou não
 - deve **restaurar** o valor dos registradores salvos antes de retornar
 - deve terminar com **IRET**

IRET retira da pilha endereço de retorno e flags

Interrupção com atendimento em endereço fixo (fixed ISR location)

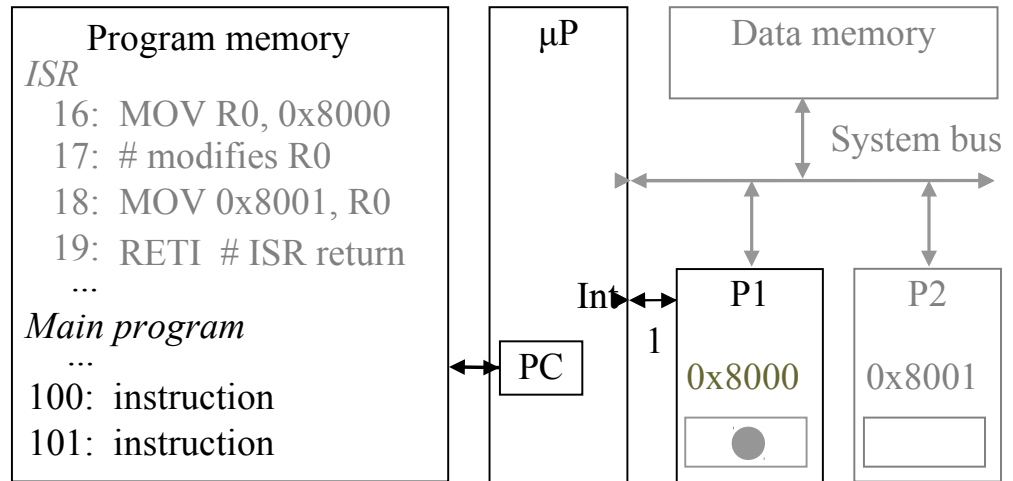
1(a): μP is executing its main program

1(b): P1 receives input data in a register with address 0x8000.



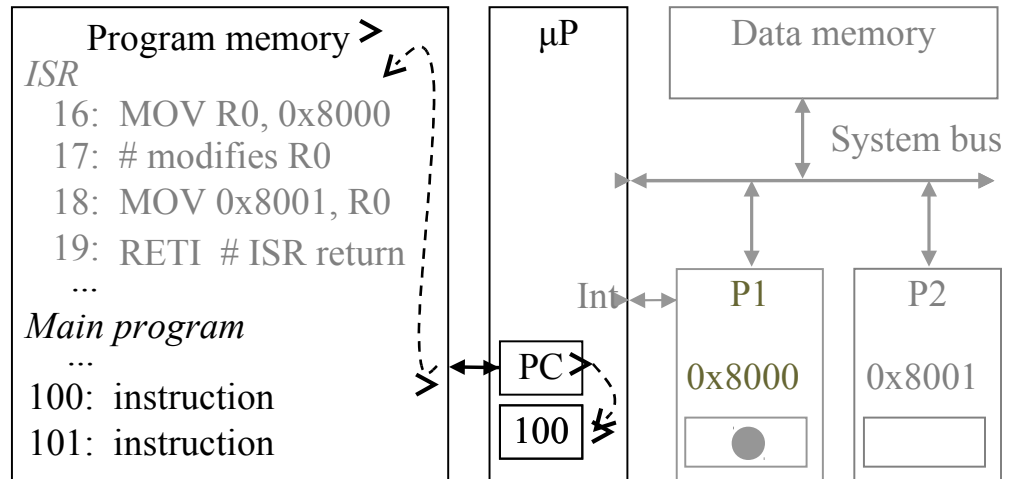
Interrupção com atendimento em endereço fixo (fixed ISR location)

2: P1 asserts *Int* to request servicing by the microprocessor



Interrupção com atendimento em endereço fixo (fixed ISR location)

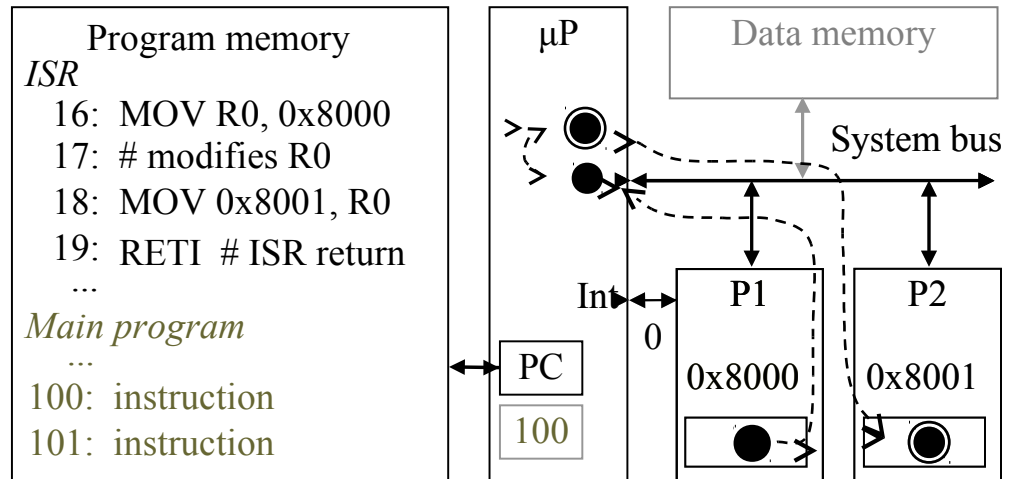
3: After completing instruction at 100, μP sees *Int* asserted, saves the PC's value of 100, and sets PC to the ISR fixed location of 16.



Interrupção com atendimento em endereço fixo (fixed ISR location)

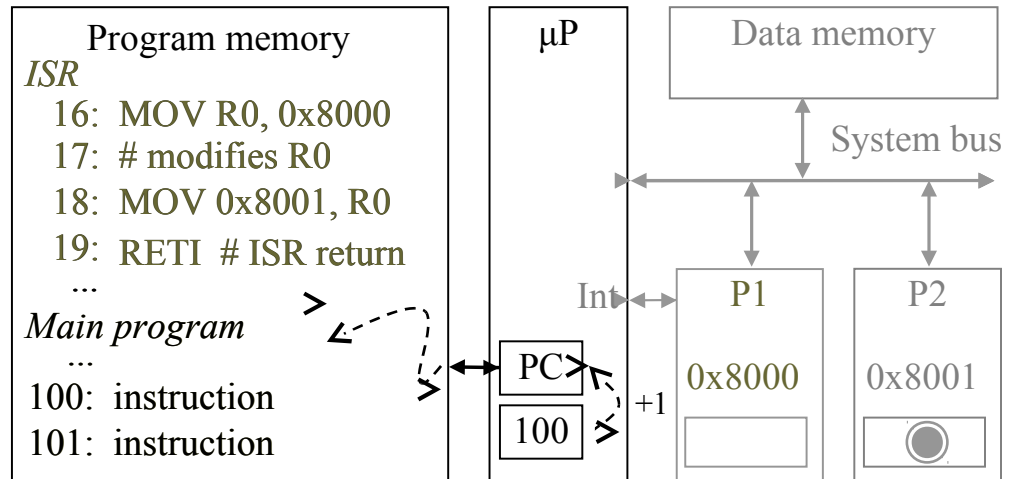
4(a): The ISR reads data from 0x8000, modifies the data, and writes the resulting data to 0x8001.

4(b): After being read, P1 deasserts *Int*.



Interrupção com atendimento em endereço fixo (fixed ISR location)

5: The ISR returns, thus restoring PC to $100+1=101$, where μP resumes executing.



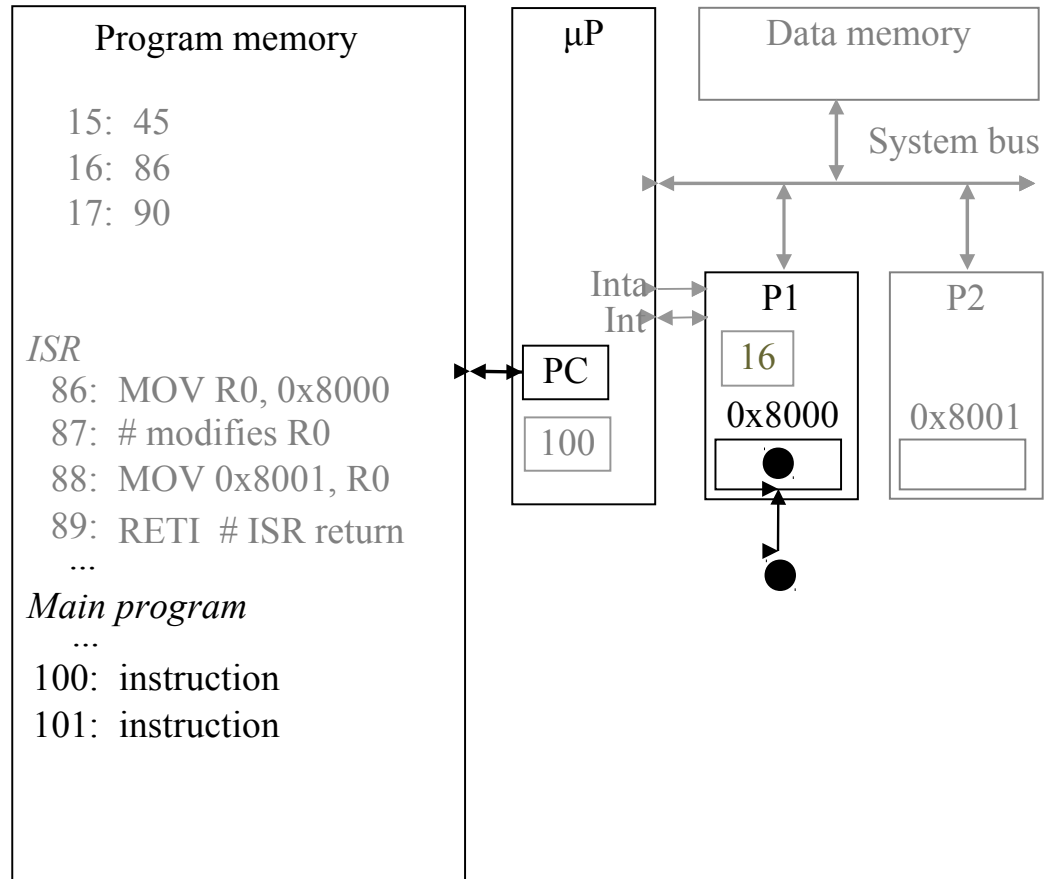
Vetores de Interrupção

- tabela de 256 posições
 - a **identificação da origem da interrupção** é usada como índice na tabela
 - *type na Instrução INT*
 - *id do periférico fornecido por um circuito externo em interrupções de hardware*
 - cada linha na tabela contém endereço de uma rotina de atendimento de interrupção (**CS:IP**)
 - no 8086 a tabela ocupa os endereços físicos de 0 a 1023
 - atualmente pode ser paginada
 - a tabela é inicializada durante boot (BIOS:modo real e SO:modo protegido)

I/O com Interrupção Vetorada

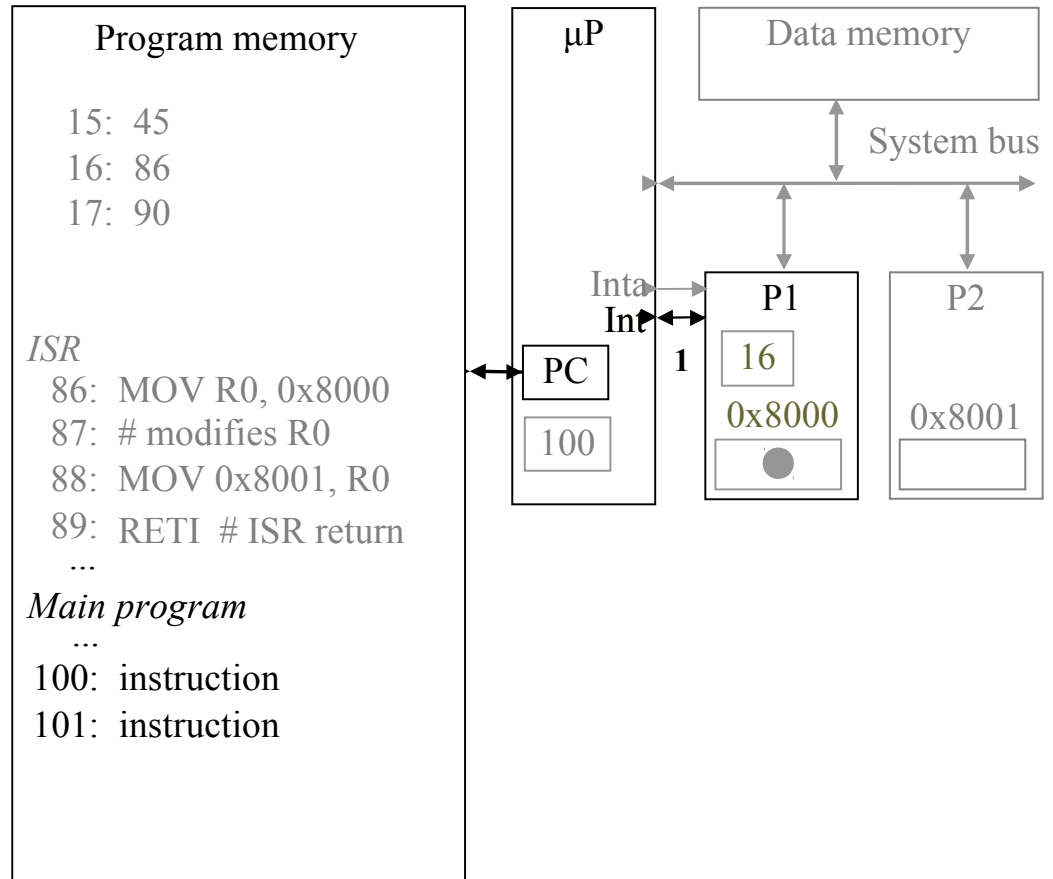
1(a): P is executing its main program

1(b): P1 receives input data in a register with address 0x8000.



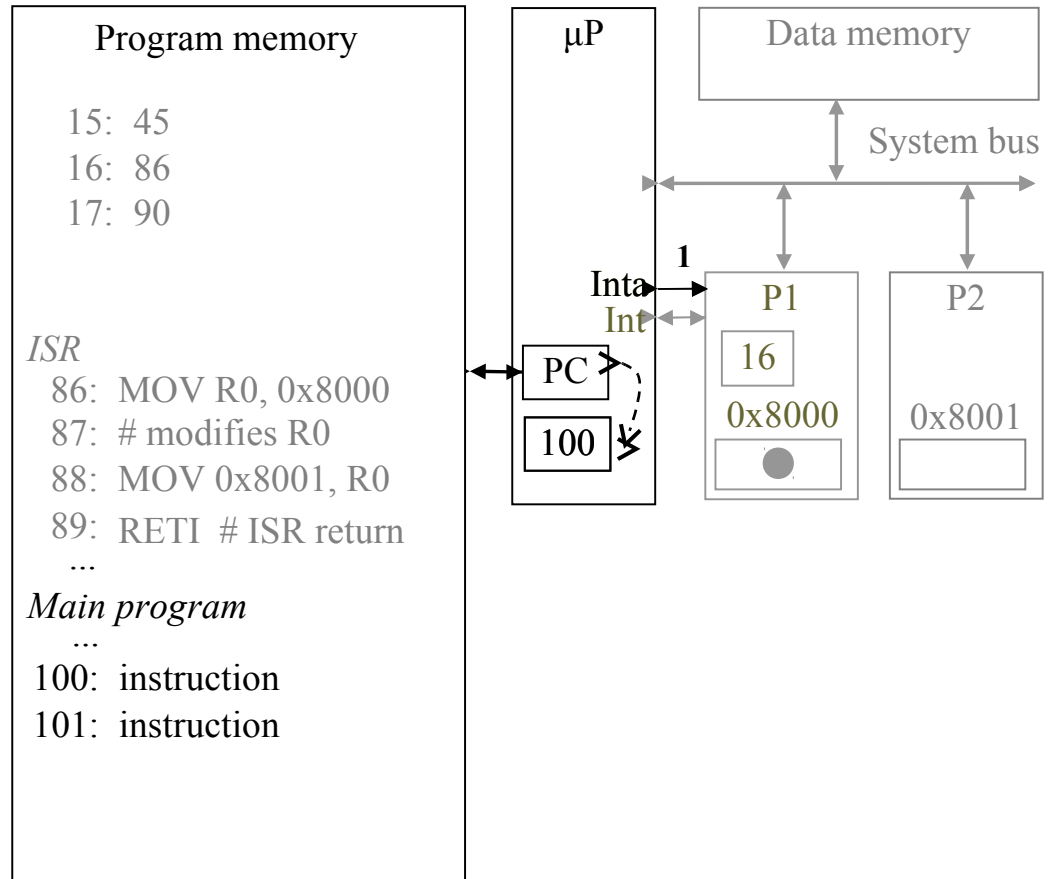
I/O com Interrupção Vetorada

2: P1 asserts *Int* to request servicing by the microprocessor



I/O com Interrupção Vetorada

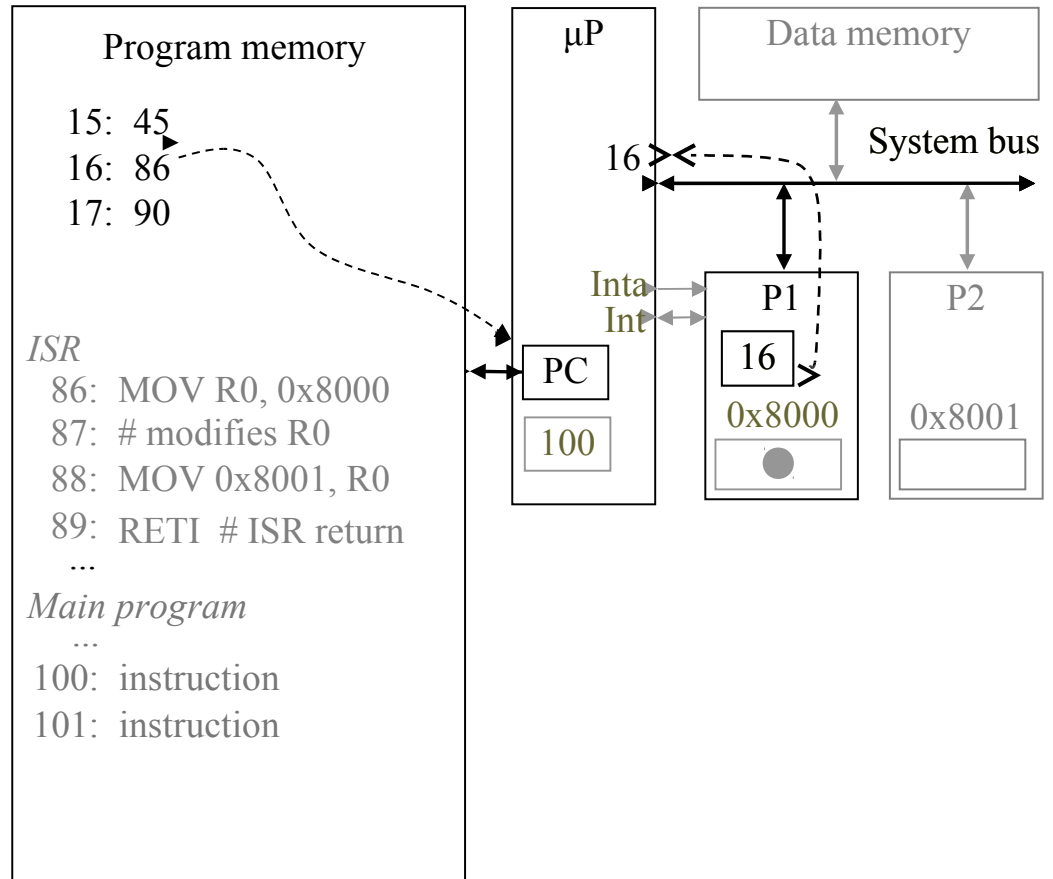
3: After completing instruction at 100, μP sees *Int* asserted, saves the PC's value of 100, and **asserts** *Inta*



I/O com Interrupção Vetorada

4: P1 detects *Inta* and puts **interrupt address vector 16** on the data bus

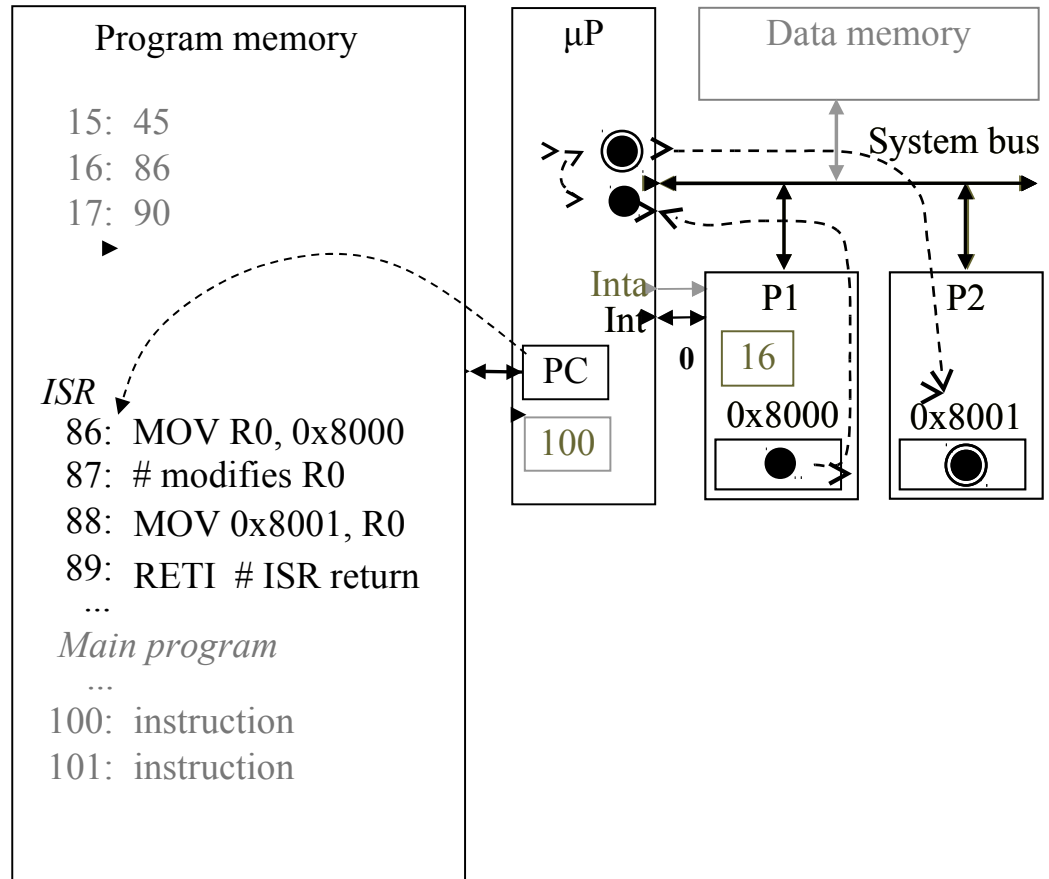
5: The value stored at address 16 will be used as PC



I/O com Interrupção Vetorada

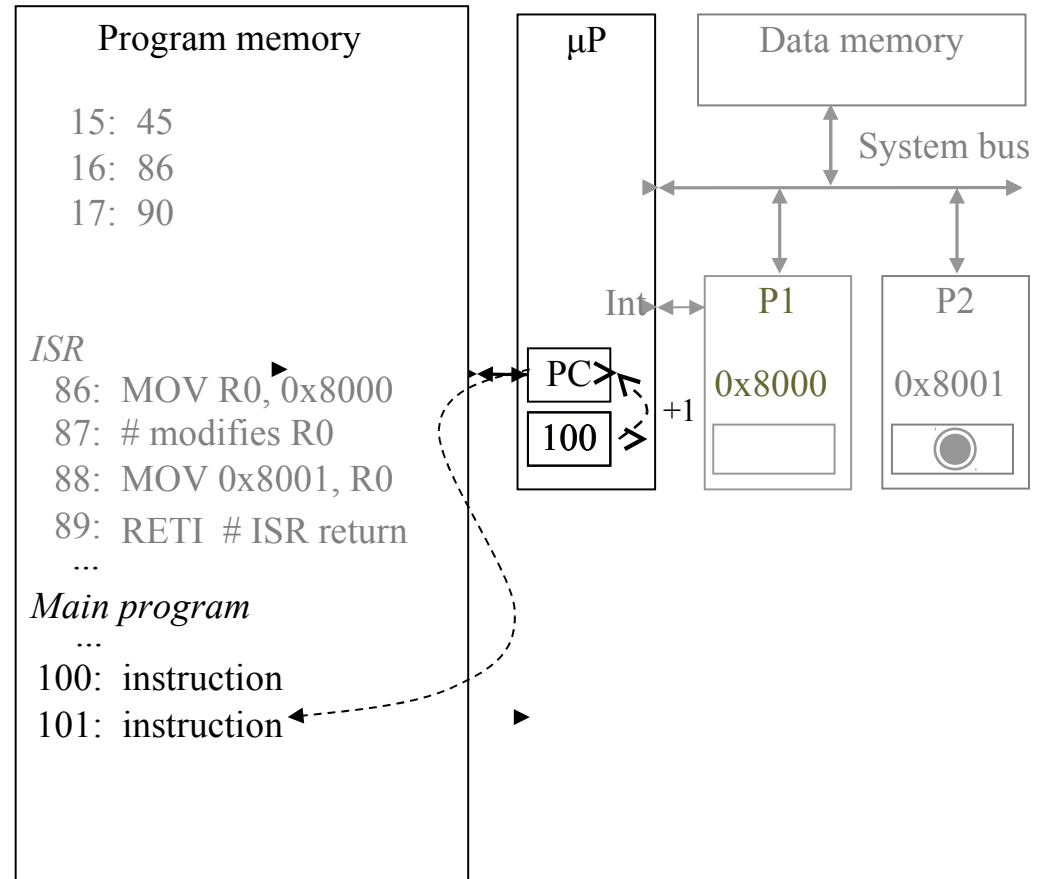
5(a): PC jumps to the address (86) stored on vector placed on the bus (16). The ISR there reads data from 0x8000, modifies the data, and writes the resulting data to 0x8001.

5(b): After being read, P1 deasserts *Int*.



Interrupt-driven I/O using vectored interrupt

6: The ISR returns, thus restoring the PC to $100+1=101$, where the μP resumes



Interrupção do microprocessador

Int.	Causa	CPU
0	divisão por zero	8086
1	execução passo a passo (DEBUG)	8086
3	ponto de parada (break point - DEBUG)	8086
4	overflow em operações aritméticas (instrução INTO)	8086
5	limite de array excedido (instrução BOUND) - a partir do 80186	80186
6	código inválido (ou código de modo protegido no modo real)	80286
7	extensão não disponível (coprocessador não presente)	80286
8	dupla interrupção (ou interrupção durante tratamento de interrupção)	80286
9	acesso fora de segmento (pelo coprocessador)	80286
10	task inválida (durante o chaveamento de ambiente)	80286
11	segmento não presente (bit de presença do segmento desligado)	80286
12	falha da pilha (overflow, underflow ou segmento de pilha não presente)	80286
13	falha geral (erros variados, não cobertos pelos casos anteriores)	80286
14	página não presente	80386
16	erro gerado pelo coprocessador	80286
17	erro de alinhamento de palavras	80486

Interrupção de Hardware

- pedido de interrupção
 - pino (**INTR**) associado a um bit de habilitação/desabilitação de interrupção (**IF**)
 - pino **NMI** para interrupções **não mascaráveis**
 - inicialmente usado para paridade de memória
- prioridade
 - prioridade entre pedidos deve ser atribuída externamente
 - é válida somente no **momento do pedido**, não no tratamento
- salvamento do contexto: **CS:IP** e **FLAGS**
 - rotina de atendimento inicia com interrupções suspensas (**IF=0**)

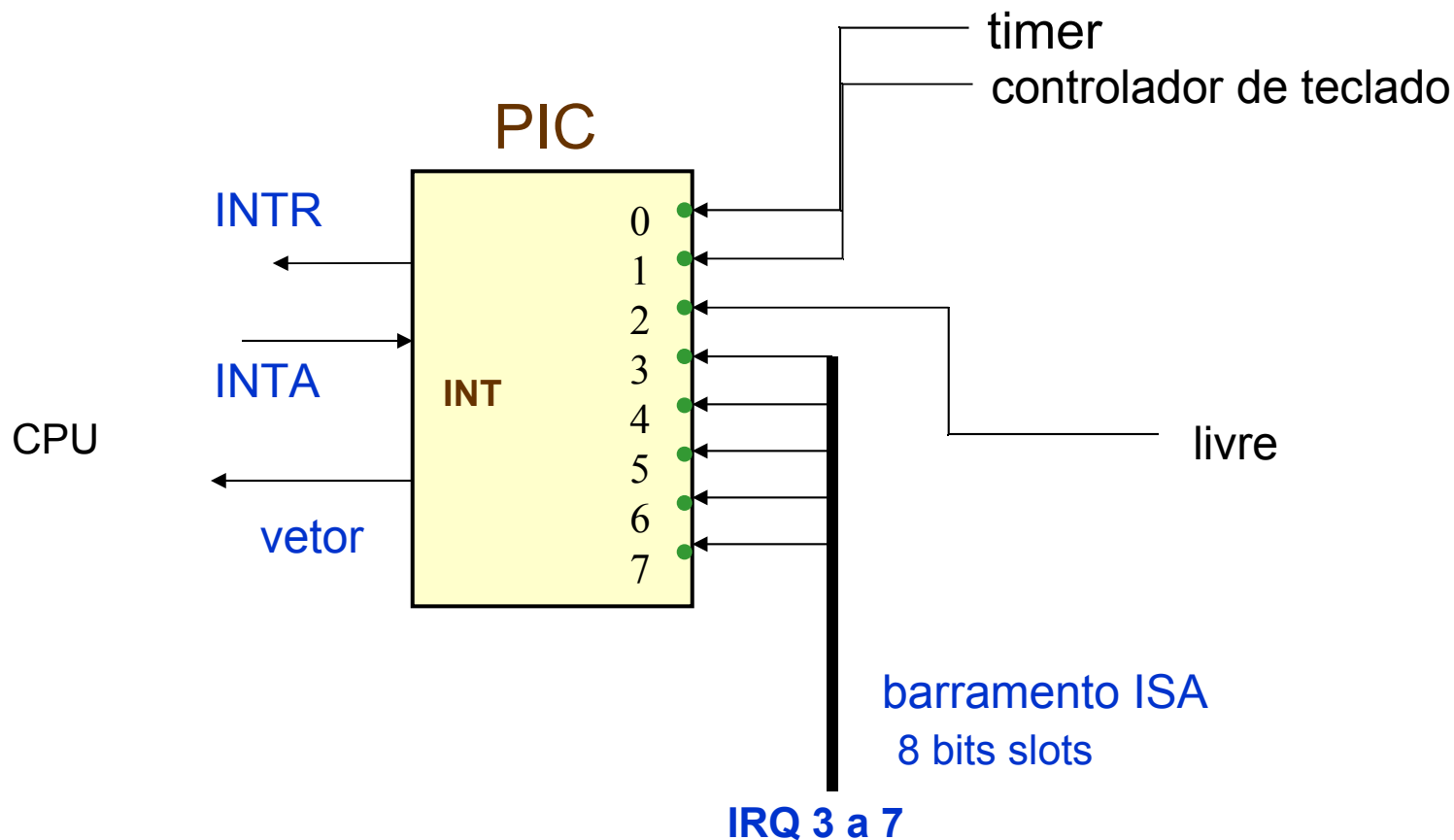
vetor 2

Interrupção de Hardware (680x0)

- Três linhas codificam o pedido de interrupção
 - IPL0, IPL1, IPL2
- Nível de operação do processador codificado em três bits
 - pedido somente é aceito se for maior que o nível de operação atual
- Atendimento:
 - salva o Status atual, altera o nível de operação (para o do pedido aceito) e chaveia para o modo supervisor
- Endereço da rotina de atendimento obtida de tabela

PIC - controlador de interrupções programável

Intel



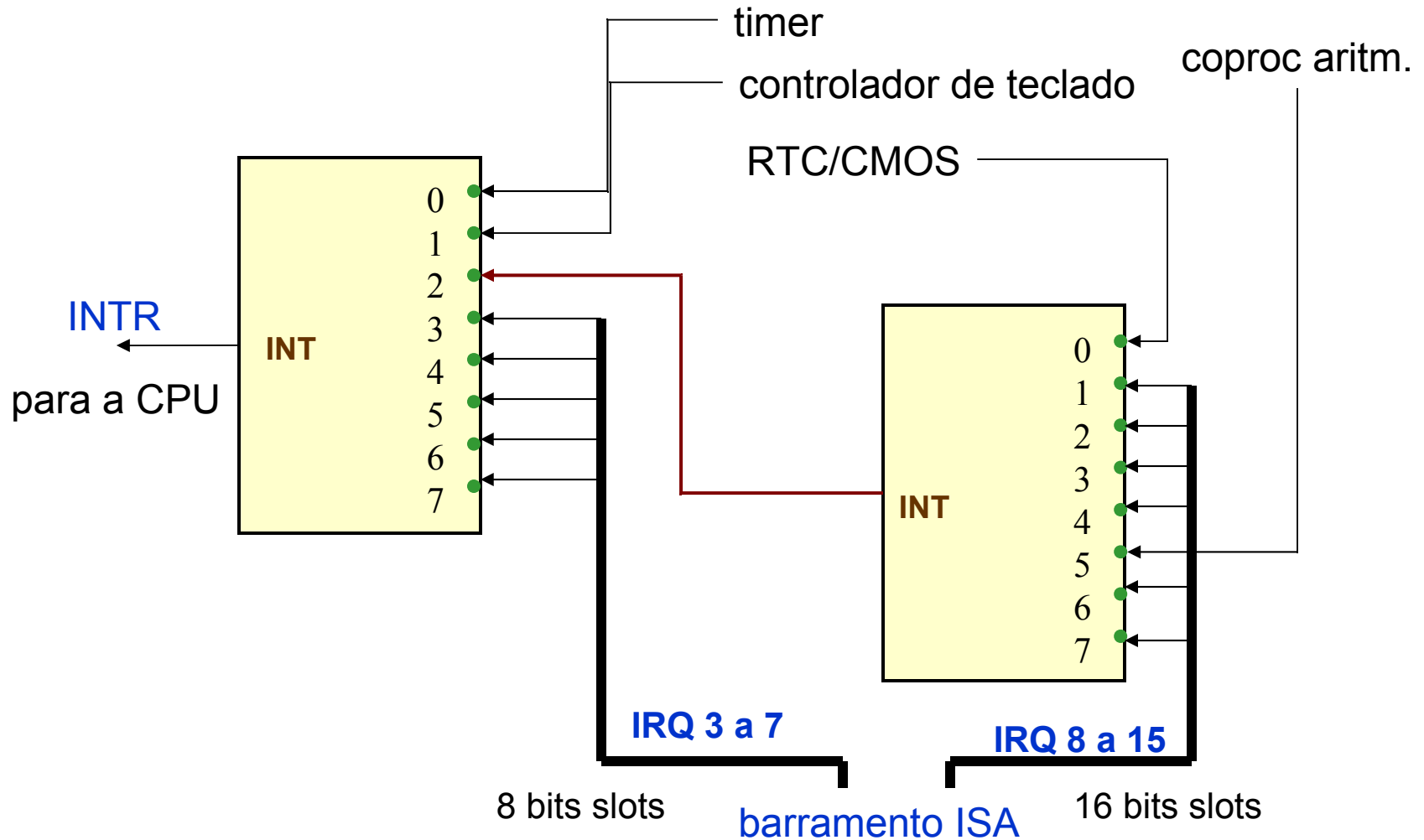
Interrupção de Hardware (XT)

barramento ISA		
Linha de Pedido	Vetor (dec.)	Utilização
IRQ0	8	relógio de tempo real (timer tick)
IRQ1	9	teclado
IRQ2	10	livre
IRQ3	11	placa serial COM2
IRQ4	12	placa serial COM1
IRQ5	13	livre (posteriormente controladora de disco rígido)
IRQ6	14	controladora de disquete
IRQ7	15	controladora de impressora (LPT1)

linhas de **IRQ** entravam num circuito externo (**PIC**) com 8 entradas e uma saída ligada o pino **INTR**


o AT usa duas PICs em cascata através da entrada **IRQ2**

PIC - controlador de interrupções programável

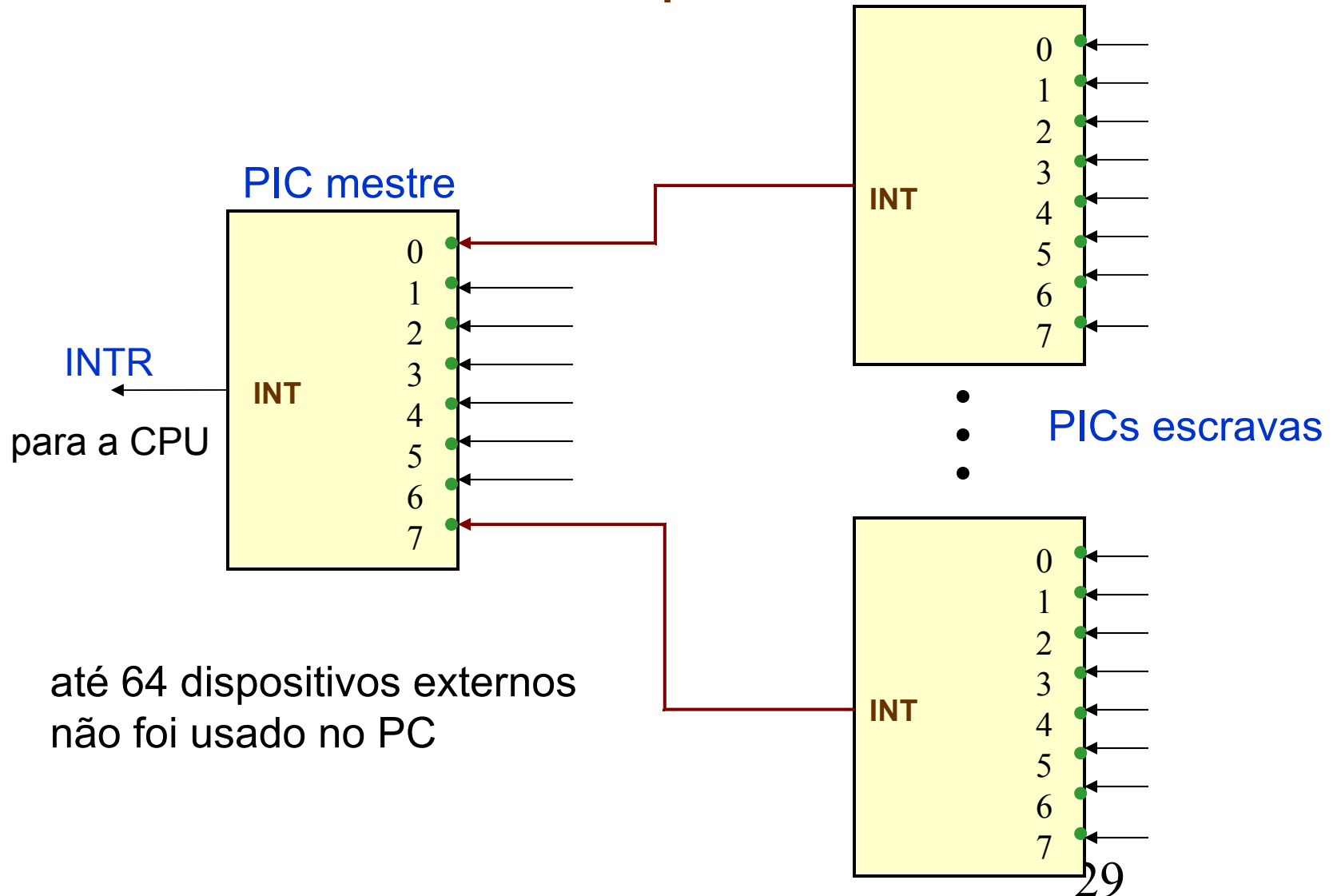


Interrupção de Hardware (AT)

barramento ISA 16

Linha de Pedido	Vetor (dec.)	Utilização
IRQ0	8	relógio de tempo real (timer tick)
IRQ1	9	teclado
IRQ2	10	acesso a PIC subordinada (cascateamento) 
IRQ8	112	atualização da hora do dia no BIOS
IRQ9	113	vídeo no PS/2, uso geral em um PC (placa de rede)
IRQ10	114	uso geral (tipicamente controlador de CD)
IRQ11	115	uso geral (tipicamente placa SCSI)
IRQ12	116	mouse no PS/2 - uso geral em um PC
IRQ13	117	coprocessador aritmético
IRQ14	118	controlador de disco rígido (controladora IDE primária)
IRQ15	119	uso geral (controladora secundária de disco)
IRQ3	11	placa serial COM2 e COM4
IRQ4	12	placa serial COM1 e COM3
IRQ5	13	uso geral (tipicamente placa de som)
IRQ6	14	controladora de disquete
IRQ7	15	controladora de impressora (LPT1)

Outro esquema



Interrupção de hardware - barramento PCI

- PCI usa compartilhamento em apenas 4 linhas
 - ISA não permite mais de um dispositivo na mesma linha de IRQ
 - **IRQA# a IRQD#**
 - **B#** a **D#** são usadas quando uma placa controladora possui mais do que um dispositivo
 - todas as controladoras com apenas um dispositivo compartilham apenas uma linha: **IRQA#**
 - compartilhamento elétrico é possível
- árbitro do barramento PCI
 - dispositivos PCI podem ser mestres e/ou escravos
 - barramento PCI é arbitrado

Interrupções PCI

- PCI IRQ Steering
 - também aparece como **IRQ steering**
 - interrupção do PCI é mapeada para as interrupções IRQn pelo chipset (ponte cpu-pci), BIOS e o sistema operacional
 - eventualmente dois dispositivos PCI podem ser mapeados para a mesma IRQx
 - geralmente cada dispositivo ganha sua própria IRQx emulada
 - plug and play
 - Windows a partir do 95
 - no Windows XP não pode ser desabilitado

Interrupção de Software

- através da instrução *INT type* no x86
- usada pelos programas para solicitar serviços ao sistema operacional
 - funciona como uma chamada de subrotina
 - mas o programa perde o controle como se tivesse sido interrompido por um dispositivo
- *system call*
 - são passados parâmetros a partir de registradores
 - algumas linguagens de programação permitem chamadas de sistema diretamente, outras indiretamente
 - os programas invocam *syscalls* em grande número e com alta frequência

Interrupção de Software (BIOS)

exemplos

Serviços do BIOS - vetores de 10H a 1FH e 40H a 5FH

- 10H - serviços de vídeo
- 11H - configuração do equipamento
- 12H - tamanho de memória
- 13H - serviços de disco
- 14H - comunicações (portas seriais RS232)
- 15H - serviços de cassete (PC) ou extensões de E/S (AT)
- 16H - serviços de teclado
- 17H - serviços de impressora
- 18H - ativação do interpretador BASIC
- 19H - tratamento de bootstrap
- 1AH - hora do dia
- 1BH - tratamento de Ctrl-Break
- 1CH - relógio de tempo real
- 67H - serviços de memória expandida

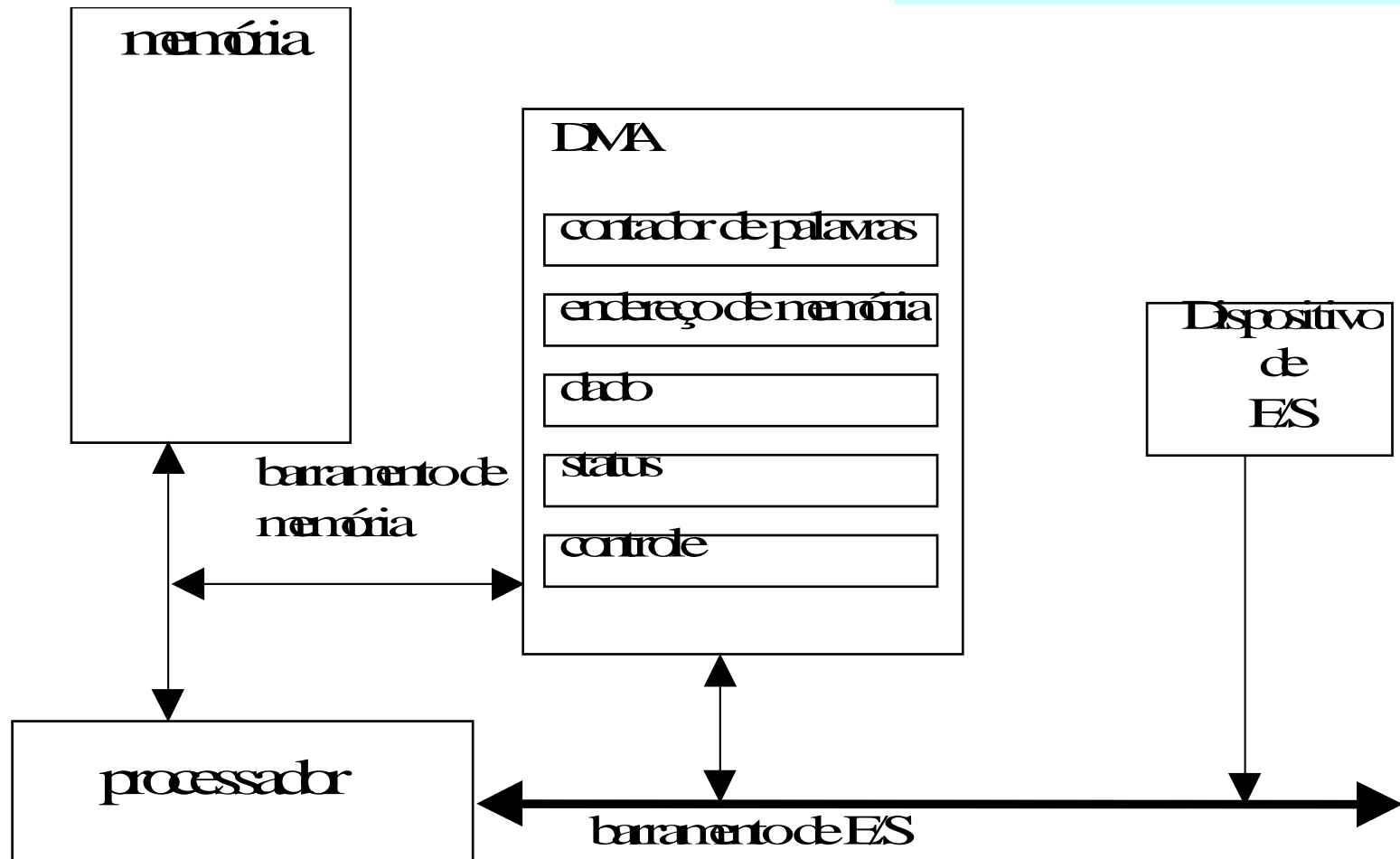
Interrupção de Software (DOS) exemplos

Serviços do DOS - vetores de 20H a 3FH

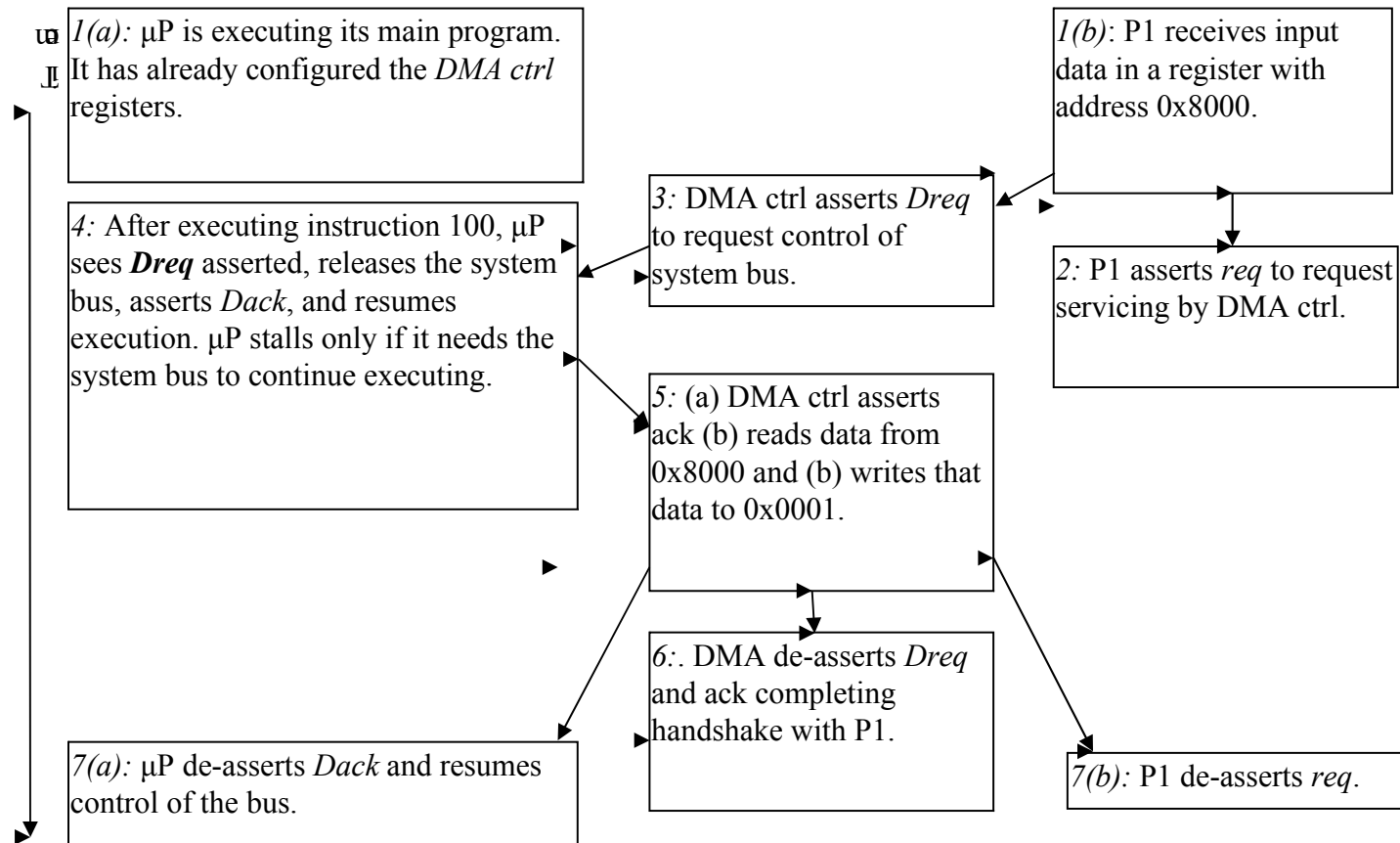
- 20H - término de programa
- 21H - serviços do DOS
- 24H - tratamento de erro
- 25H - leitura absoluta de disco
- 26H - escrita absoluta de disco
- 27H - término de programa residente
- 2AH - serviços de rede DOS
- 2EH - comando de execução DOS
- 2FH - controle de spooler de impressão DOS
- 33H - serviços do driver de mouse

Acesso Direto a Memória (DMA)

esquema convencional



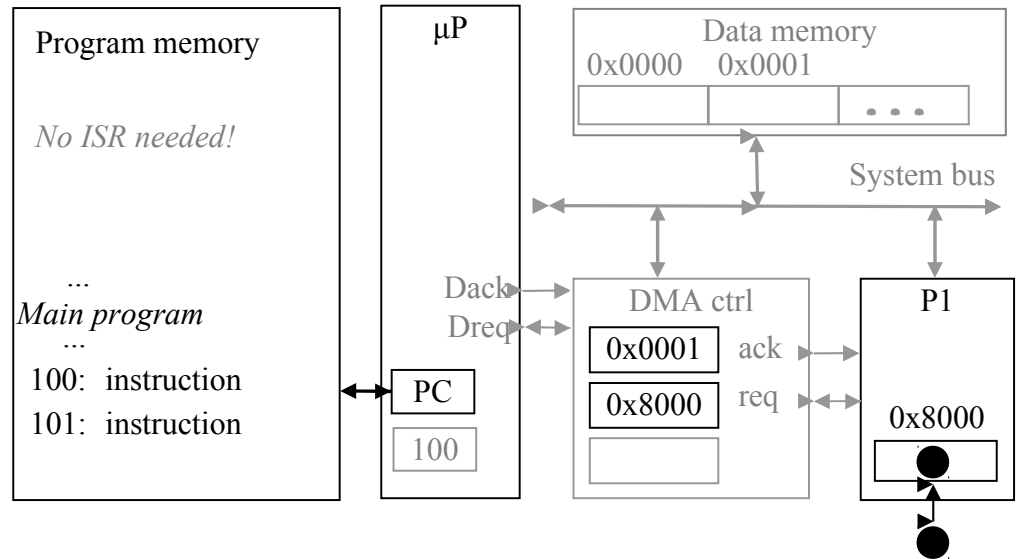
I/O para memória usando DMA



I/O para memória usando DMA(cont')

1(a): μP is executing its main program. It has already configured the DMA ctrl registers

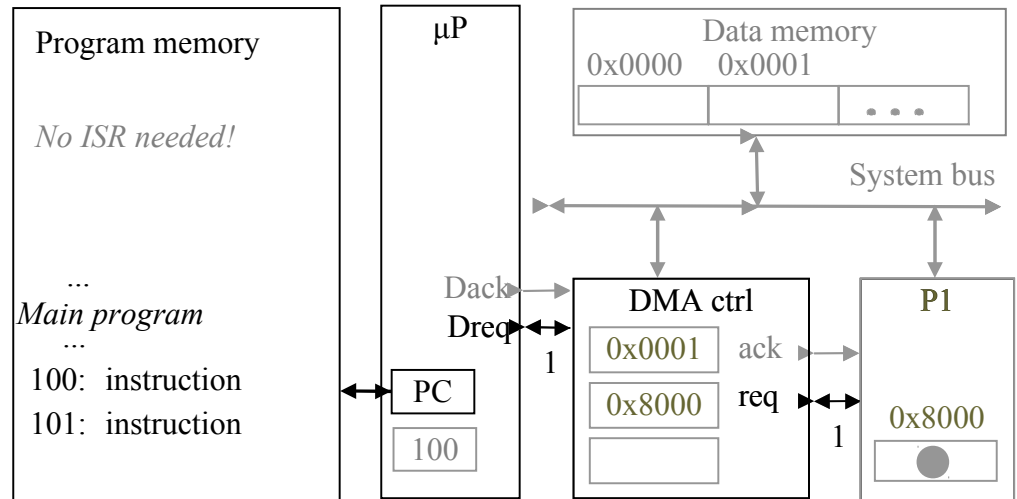
1(b): P1 receives input data in a register with address 0x8000.



I/O para memória usando DMA(cont')

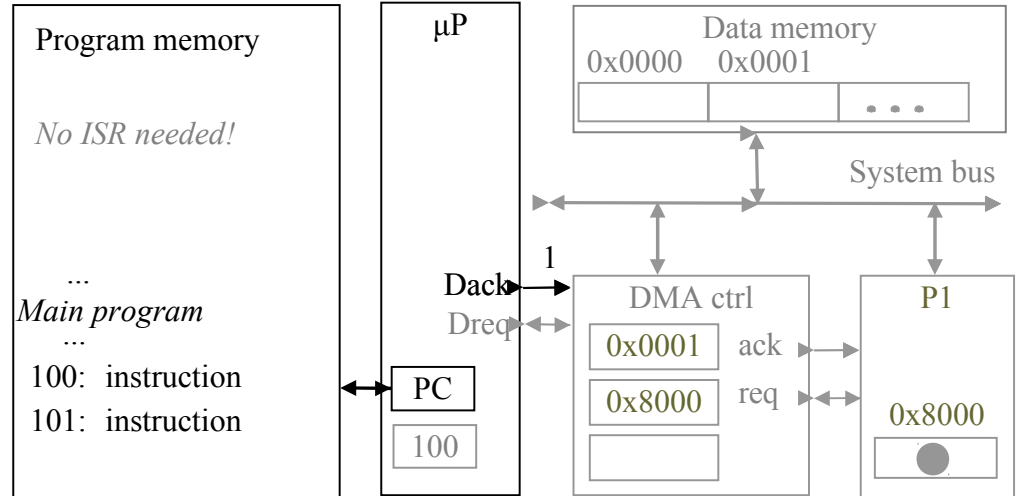
2: P1 asserts *req* to request servicing by DMA ctrl.

3: DMA ctrl asserts *Dreq* to request control of system bus



I/O para memória usando DMA(cont')

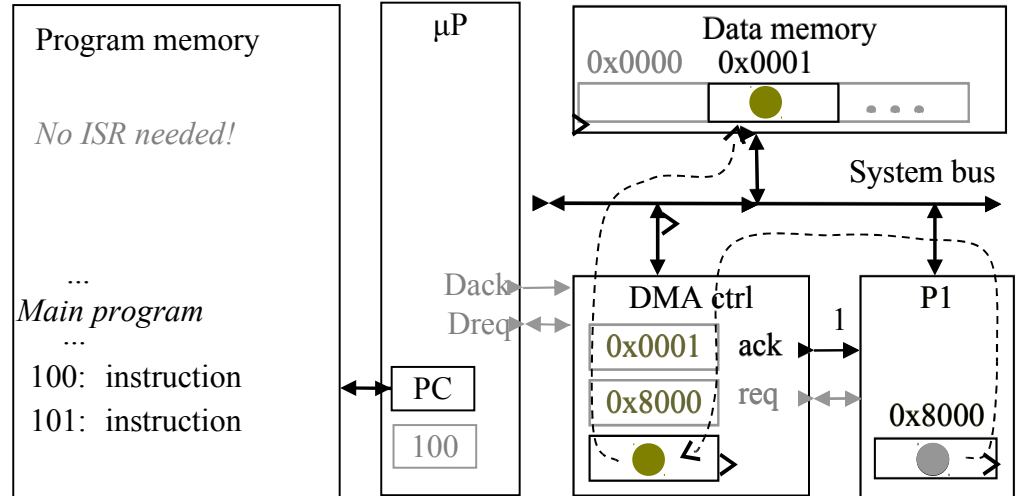
4: After executing instruction 100, μP sees *Dreq* asserted, releases the system bus, asserts *Dack*, and resumes execution, μP stalls only if it needs the system bus to continue executing.



I/O para memória usando DMA(cont')

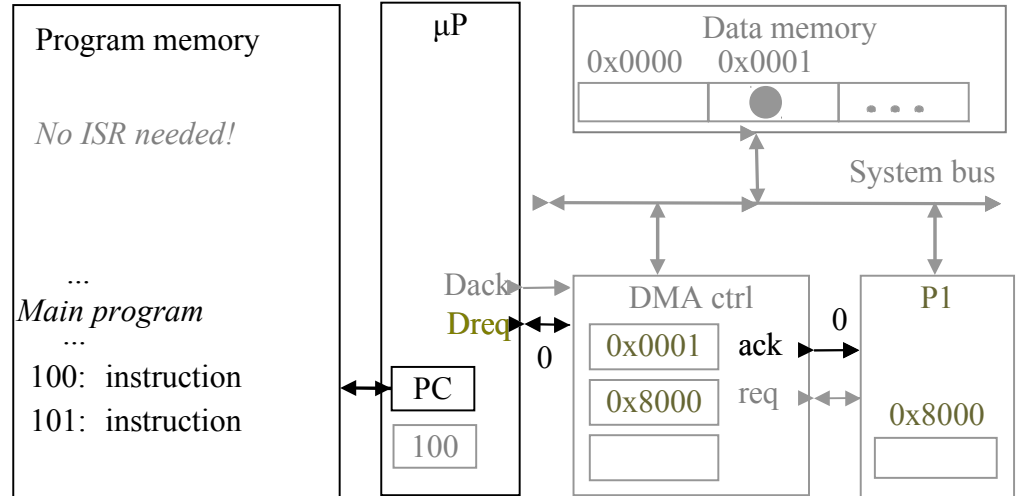
5: DMA ctrl (a) asserts ack, (b) reads data from 0x8000, and (c) writes that data to 0x0001.

(Meanwhile, processor still executing if not stalled!)



I/O para memória usando DMA(cont')

6: DMA de-asserts *Dreq* and *ack* completing the handshake with P1.



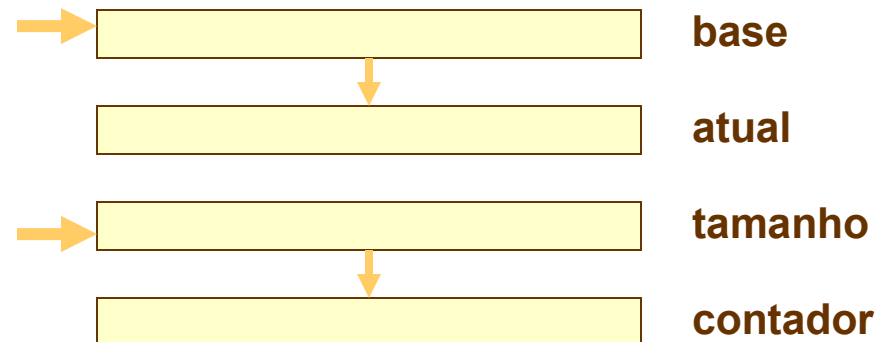
Controlador de DMA (original)

barramento ISA

- controlador de DMA 8237, operando a 5 MHz
 - 4 canais de DMA independentes
 - o tamanho máximo do bloco a ser transferido é de 64 KB
 - cada canal possui 1 registrador de 6 bits para controle e 4 registradores de 16 bits

- registradores de um canal

- endereço atual
- contador de bytes atual
- endereço base
- tamanho do bloco



Controlador de DMA (original)

barramento ISA

- registradores do controlador
- 4 registradores de controle comuns a todos os canais
 - registrador de mascaramento de canal (individual)
 - registrador de requisição de canal
 - registrador básico de comando
 - registrador de status

Controlador de DMA (original)

barramento ISA

- IBM PC original:
 - canal 0: refresh de memória
 - canal 2: operações de disco (disquete)
 - canais 1 e 3: disponíveis
- transferências **memória a memória**
 - 2 canais de DMA
 - obrigatoriamente o canal 0 deve participar
- 8237 pode endereçar apenas uma memória de **64 KBytes**
 - para acesso à 1 MB, o PC usava um conjunto de 4 registradores de 4 bits, um para cada canal, que completam o endereço para 20 bits

DMA hoje

- PCs atuais
 - **emulam** o funcionamento dos 8 canais de DMA (chipset)
 - permitem que todos os canais manipulem dados de 16 bits
- refresh de memória
 - passada para um circuito específico (*concurrent refresh*, ou *hidden refresh*, ou *self-refresh*)
 - liberação do canal 0 e dispensa da CPU de ativar o refresh (cerca de **10%** do tempo de processamento era consumido com o controle do refresh)

DMA hoje: canais

Canal de DMA	Dados	Uso típico
0	8 ou 16 bits	Audio
1	8 ou 16 bits	Audio ou rede
2	8 ou 16 bits	Floppy
3	8 ou 16 bits	Porta paralela (ECP ou EPP)
4	16 bits	canal de cascadeamento
5	16 bits	livre
6	16 bits	livre
7	16 bits	IDE ISA