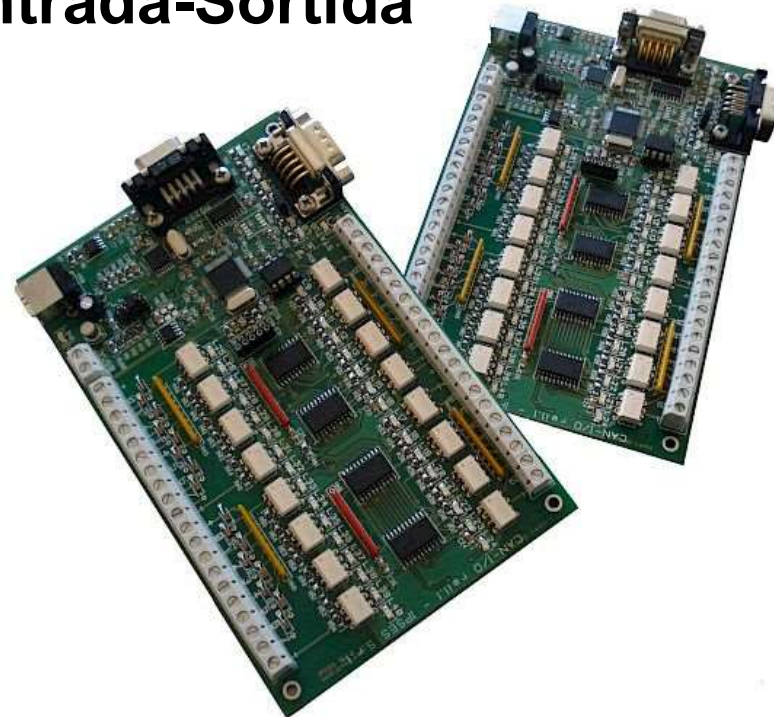
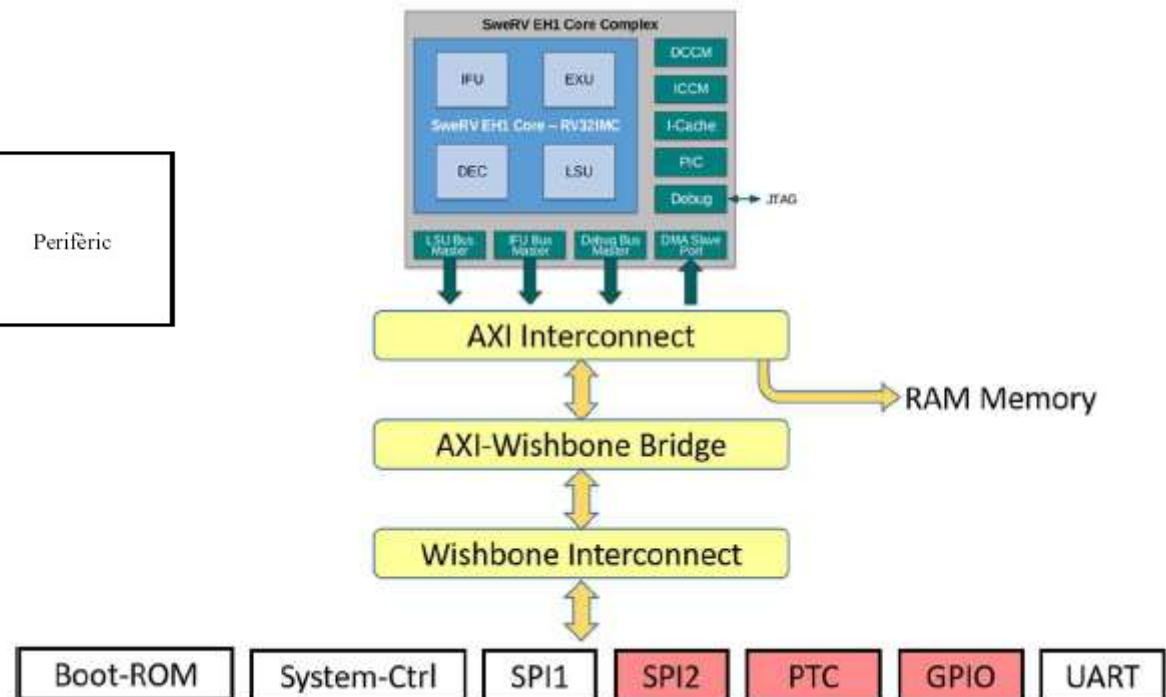
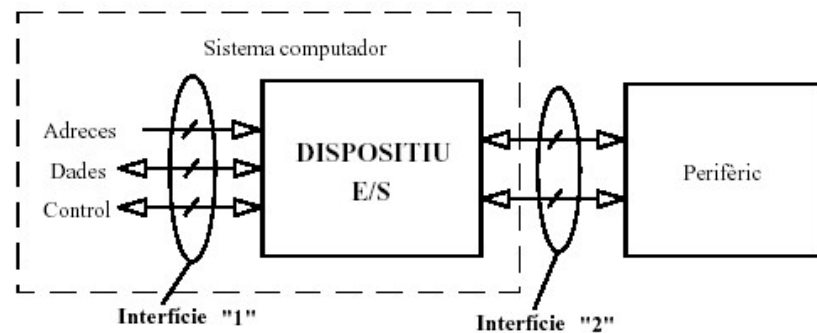


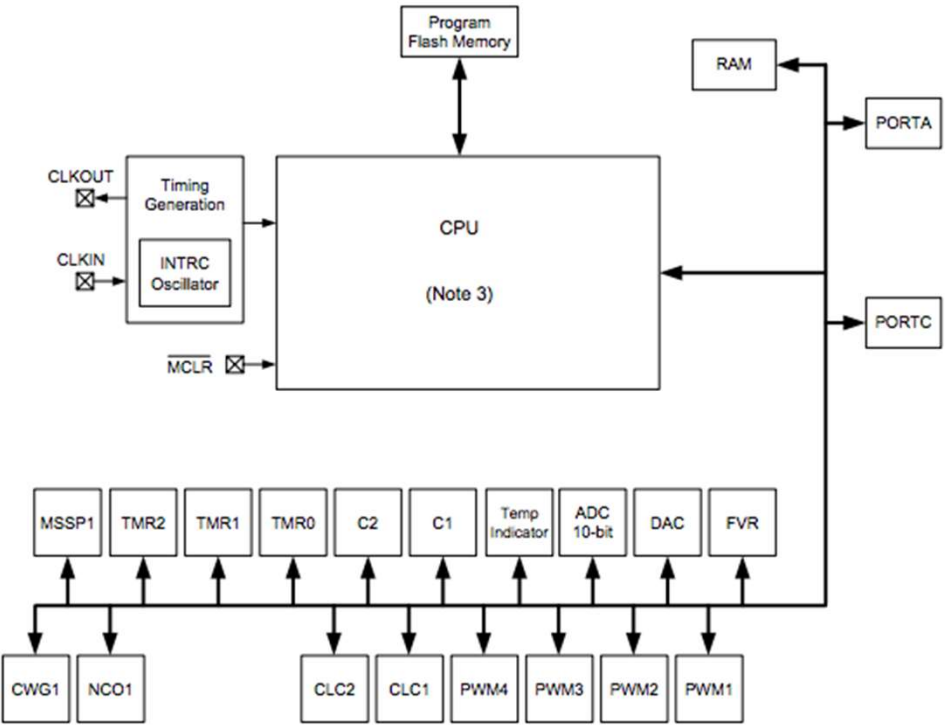
Interfícies d'Entrada-Sortida



El dispositiu d'entrada sortida ens permet **ESTABLIR LA COMUNICACIÓ DE LA NOSTRA CPU AMB L'EXTERIOR**



Exemple de dispositiu



Microcontrolador

TABLE 1-1: DEVICE PERIPHERAL SUMMARY

Peripheral	PIC12(L)/F1501	PIC16(L)/F1503	PIC16(L)/F1507	PIC16(L)/F1508	PIC16(L)/F1509
Analog-to-Digital Converter (ADC)	•	•	•	•	•
Complementary Wave Generator (CWG)	•	•	•	•	•
Digital-to-Analog Converter (DAC)	•	•		•	•
Enhanced Universal Synchronous/Asynchronous Receiver/Transmitter (EUSART)				•	•
Fixed Voltage Reference (FVR)	•	•	•	•	•
Numerically Controlled Oscillator (NCO)	•	•	•	•	•
Temperature Indicator	•	•	•	•	•
Comparators					
	C1	•	•	•	•
	C2		•	•	•
Configurable Logic Cell (CLC)					
	CLC1	•	•	•	•
	CLC2	•	•	•	•
	CLC3			•	•
	CLC4			•	•
Master Synchronous Serial Ports					
	MSSP1		•	•	•
PWM Modules					
	PWM1	•	•	•	•
	PWM2	•	•	•	•
	PWM3	•	•	•	•
	PWM4	•	•	•	•
Timers					
	Timer0	•	•	•	•
	Timer1	•	•	•	•
	Timer2	•	•	•	•

Interfícies d'entrada sortida

Peripherals -- Any piece of hardware that isn't mounted inside a PC's case is called a peripheral. This includes your basic input and output devices: monitors, keyboards and mice. It also includes printers, speakers, headphones, microphones, webcams and USB flash drives. Anything you can plug in to a port on the PC is one of the PC's peripherals. The essential peripherals (such as monitors) aren't necessary on laptops, which have them built in instead.

Expansion slots -- On occasion, you'll want to add components to a PC that don't have a designated slot somewhere on the motherboard. That's why the motherboard will include a series of expansion slots. The removable components designed to fit into expansion slots are called cards, probably because of their flat, card-like structure. Using expansion slots, you can add extra video cards, network cards, printer ports, TV receivers and many other custom additions. The card must match the expansion slot type, whether it's the legacy ISA/EISA type or the more common [PCI](#), PCI-X or PCI Express types.

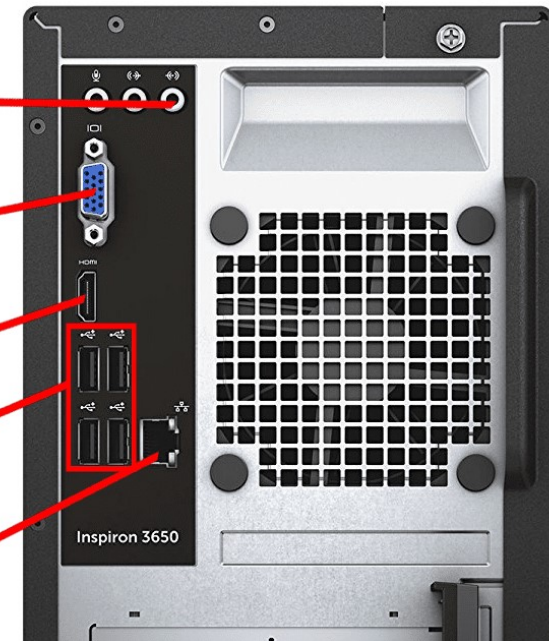
Audio Ports

VGA Port

HDMI Port

USB Ports

LAN Port



Peripherals connected to a computer need special communication links for interfacing with CPU. In computer system, there are special hardware components between the CPU and peripherals to control or manage the input-output transfers. These components are called **input-output interface units** because they provide communication links between processor bus and peripherals. They provide a method for transferring information between internal system and input-output devices.

Però abans, **RECORDEM:**

El microprocessador accedeix a la memòria on tenim:

- La memòria de programa
- La memòria de dades
- La memòria assignada a les interfícies d'E/S

El procés de transferència entre CPU i I/O requereix una gestió.
Gestió realitzada per la CPU o pel dispositiu E/S

Un element determinat d'E/S té com a interfície amb la CPU una o diverses adreces de memòria on pot depositar informació o la pot exteure.

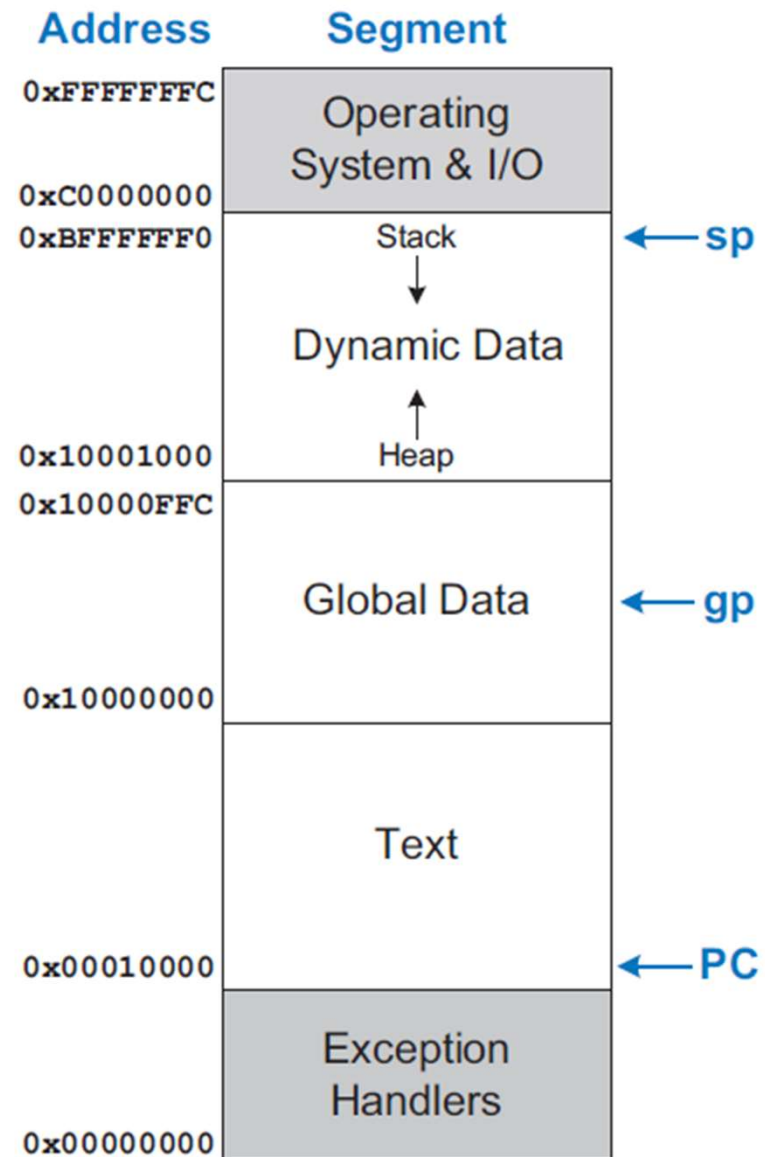
In RISC-V a portion of the address space is dedicated to I/O devices rather than memory.

For example, suppose that physical addresses in the range 0x20000000 to 0x20FFFFFFF are used for I/O. Each I/O device is assigned one or more memory addresses in this range.

A store to the specified address sends data to the device.

A load receives data from the device. This method of communicating with I/O devices is called:

memory-mapped I/O



La informació que s'envia al perifèric pot ser de dos tipus:

- Dades, enviades o rebudes o bé...

- Senyals de control de la interfície

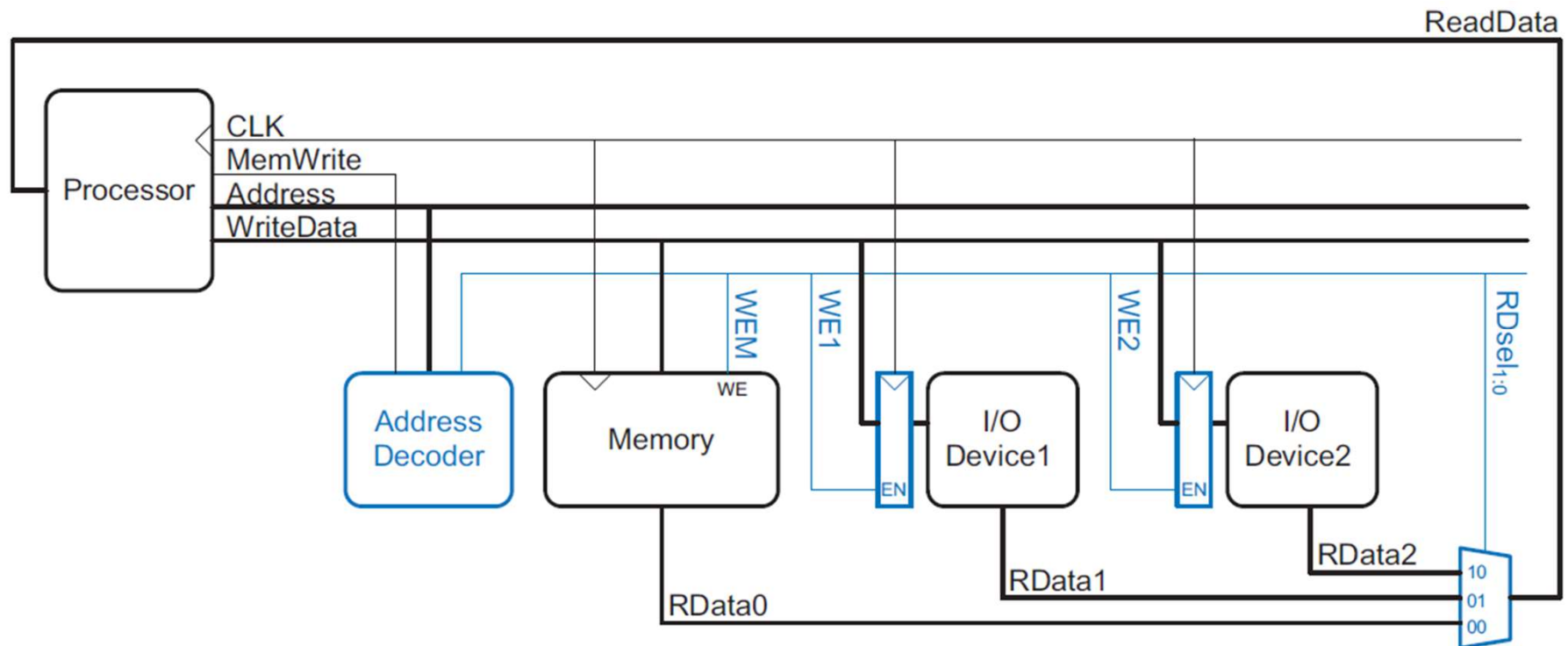
- Read/Write

- Chip Select

- Ready

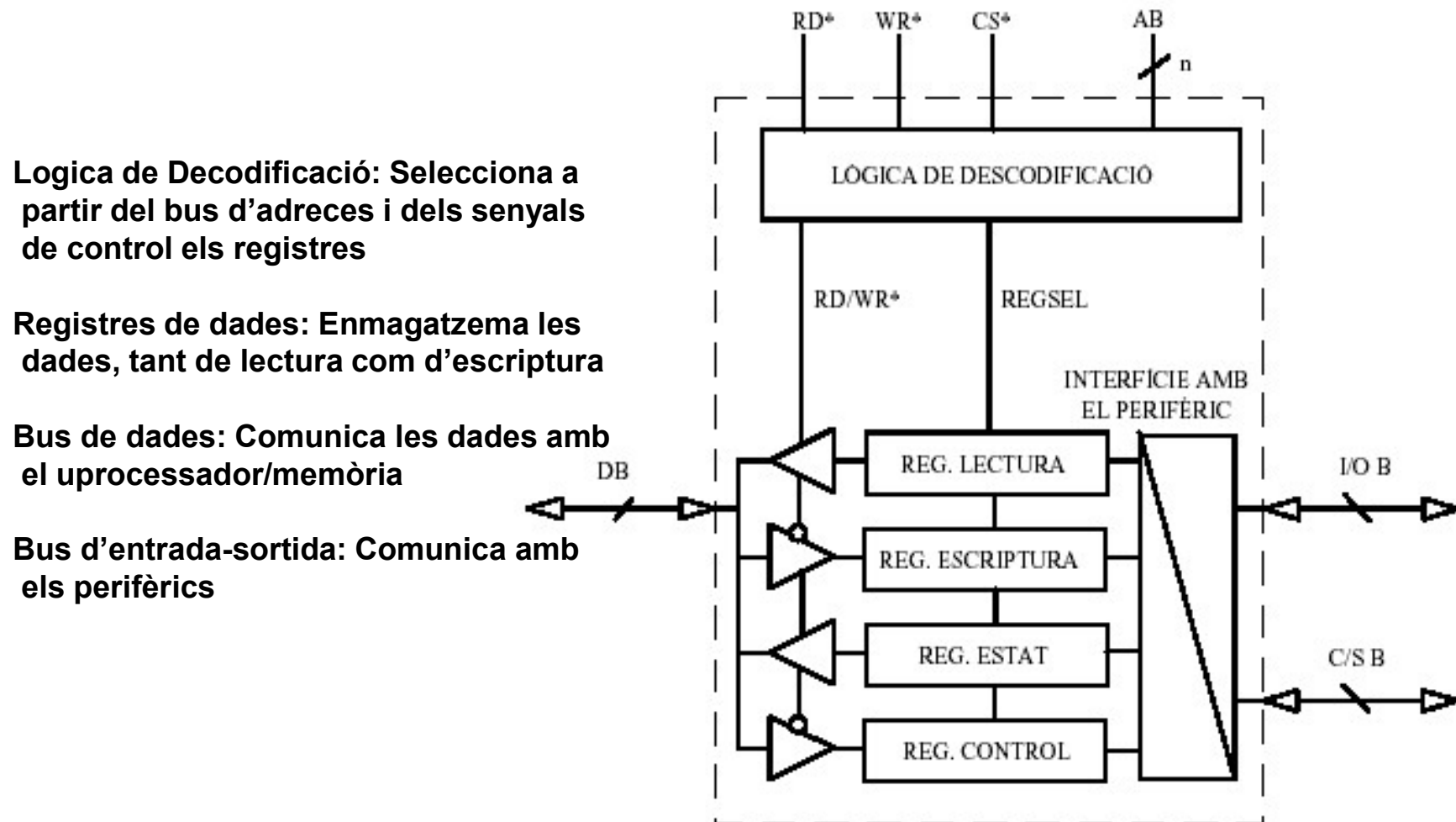
- ...

- Les posicions de memòria utilitzades per realitzar l'intercanvi d'informació reben el nom de registres de Dades.
- Depenent del sentit de la informació seran dades de lectura o escriptura



hardware needed to support two memory-mapped I/O devices. An address decoder determines which device communicates with the processor.

L'estructura de blocs d'un dispositiu d'E/S queda doncs de la següent forma



Els dispositius d'E/S han de soportar les següents funcions:

- 1.- Control i sincronització en la transferència de dades entre el processador i la memòria principal amb els perifèrics
- 2.- Establiment del camí de comunicació entre els recursos interns i els perifèrics
- 3.- Detecció i manipulació d'errors
- 4.- Enmagatzament temporal de les dades

EL PROBLEMA DE LA SINCRONITZACIÓ

El funcionament del processador es fa amb un cert sincronisme que ve relacionat amb el rellotge al qual estigui connectat.

Malauradament (com a mínim per al microprocessador) la vida real es ASINCRONA per tant s'haurà de produir una certa adaptació entre els dos sistemes. Ej: teclat i pantalla

Si el que volem es transferir dades a la pantalla, el uP només s'ha d'adreçar al controlador en qüestió i demanar-li si està preparat per rebre les dades. Si està disponible li envia, si no, s'esperarà fins que la pantalla estigui disponible.

Encara que pugui trigar “molt de temps” l'usuari final no se n'adona

EL PROBLEMA DE LA SINCRONITZACIÓ (II)

El problema es planteja quan és el perifèric (teclat) qui demana l'atenció del uP.

- El perifèric no pot controlar cap mena de transferència ni utilitzar el bus del sistema quan a ell li sembli oportú
- Només pot cridar l'atenció del uP per tal que aquest realitzi la seva demanda quan ho consideri oportú
- El sistema podria estar eternament demanant l'atenció del uP i aquest no fer-li cap mena de cas...

COM ES POT SOL.LUCIONAR AQUEST PROBLEMA??

El uP ha de conèixer de l'existència del perifèric i dedicar-li atenció d'alguna manera per tal de gestionar la transferència de informació.

Existeixen dos (+1) tipus de sistemes de gestió de transferència de informació:

1.- POLLING o consulta: El control l'assumeix el uP


2.- Interrupcions: El control vindrà assumit per al dispositiu d'E/S

...i 3.- El DMA...on entrarem més endavant

POLLING

El uP ha de controlar PERIODICAMENT l'estat dels perifèrics a partir dels controladors d'E/S.

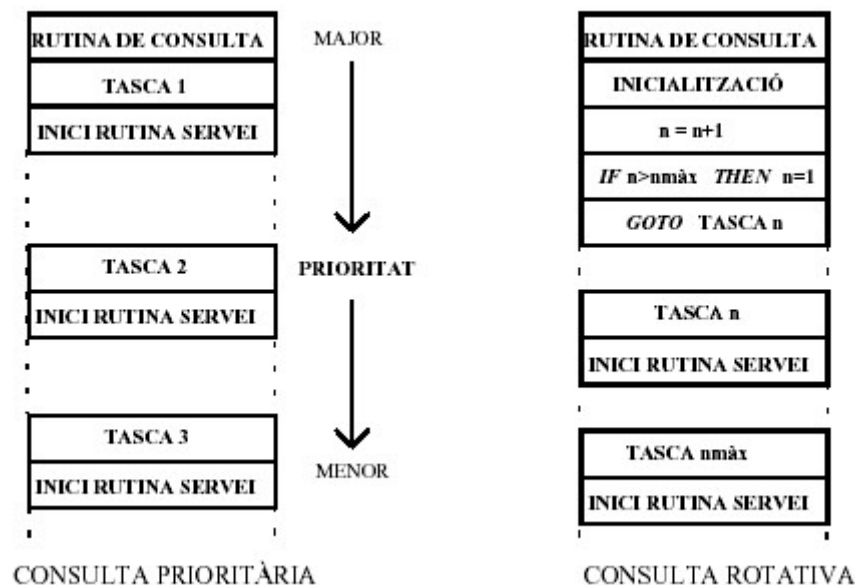
Al programa tenim una subrutina que s'executarà periòdicament que s'encarrega de llegir els registres d'estat corresponents a cada perifèric

Cada cert temps sincronitzem els dos sistemes  sincronització per consulta o Polling

La sincronització per consulta es un procés de SOFTWARE

La freqüència de mostreig és en aquest punt fonamental per tal de no perdre cap mena d'informació (teorema del mostreig)

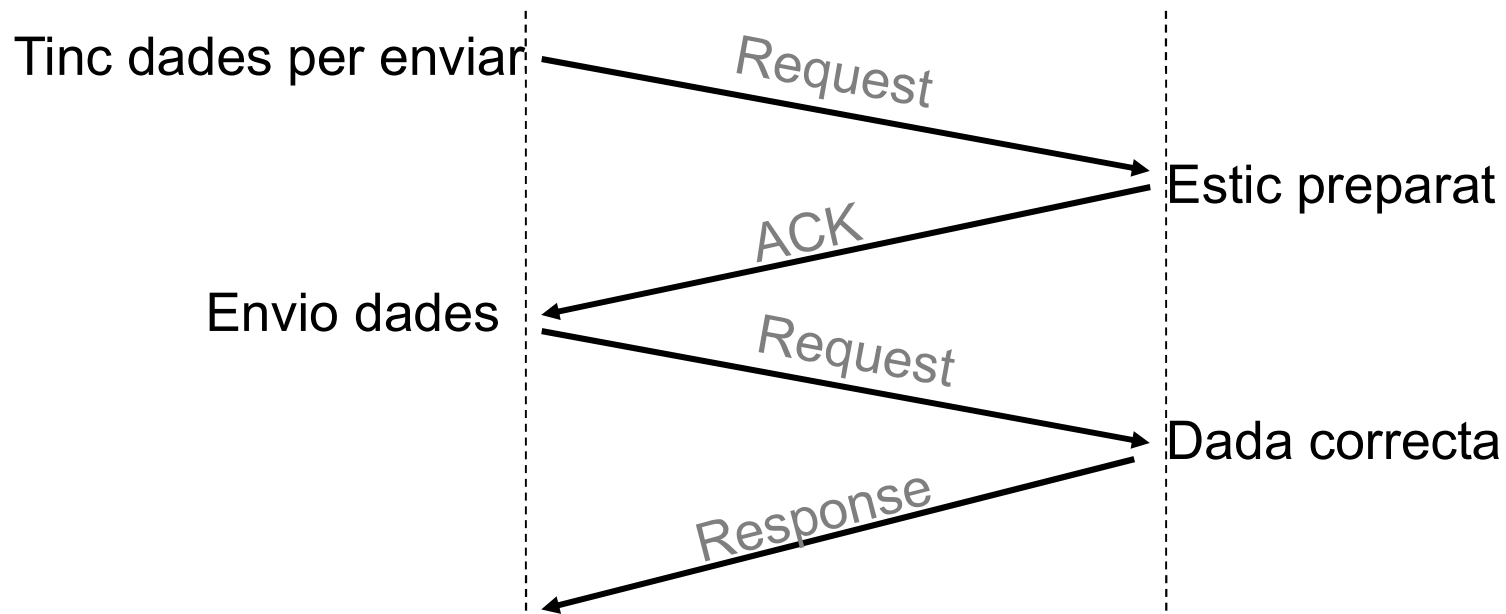
El programador pot establir una certa prioritat de consultes als perifèrics



Comunicació CPU/Perifèric controlada per CPU:

-Els dos han d'estar preparats per $\left\{ \begin{array}{c} \text{transmetre} \\ \text{rebre} \end{array} \right\}$ informació

HANDSHAKING:



Exemple 2:

A la posició FFh del mapa de E/S del RISCV hi ha associat un controlador que adapta una impressora a la que es vol enviar per imprimir els caràcters ASCII que es troben carregats en les adreces 0, 1 i 2 del mapa de memòria. Confeccioneu el programa que realitzi aquesta transferència d'informació desde la memòria principal al controlador del perifèric

Sol.lució:

Per treure el caràcter que es desitja imprimir previamente s'haurà de portar a la posició FFh del mapa E/S.

Per tant s'ha de carregar un registre amb el contingut de la posició 0 del mapa de memòria

```
lw a0, 0(zero)
sw a0, 0xFF(zero)
lw a0, 4(zero)
sw a0, 0xFF(zero)
lw a0, 8(zero)
sw a0, 12(zero)
```

En aquest exemple hem considerat que la impresora es tant ràpida com el uP

En la pràctica el processador abans d'enviar una nova dada a la impresora llegeix el registre d'estat del controlador per coneixer si ha terminat d'imprimir el caràcter anterior

```
lw a0, 0(zero)
```

```
Delay(N)
```

```
sw a0, 0xFF(zero)
```

```
Delay(N)
```

```
lw a0, 4(zero)
```

```
Delay(N)
```

```
sw a0, 0xFF(zero)
```

```
Delay(N)
```

```
lw a0, 8(zero)
```

```
sw a0, 12(zero)
```

Suppose that I/O Device 1 in [Figure PP8](#) assigned the memory address 0x20001000. Show the RISC-V assembly code for writing the value 7 to I/O Device 1 and for reading the output value from I/O Device 1.

Solution The following RISC-V assembly code writes the value 7 to I/O Device 1. The `.equ` assembler directive replaces the named symbol with the given value. So, the `li s1, ioadr` instruction becomes `li s1, 0x20001000`.

```
.equ ioadr    0x20001000

    li s0, 7
    li s1, ioadr
    sw s0, 0(s1)
```

The address decoder detects address 0x20001000 and *MemWrite* = 1, so it asserts *WE1*, the write enable for Device 1's register. At the next clock edge, the value on the *WriteData* bus, 7, is written into the register, whose output connects to the input pins of I/O Device 1.

To read from I/O Device 1, the processor executes the following RISC-V assembly code.

```
lw s0, 0(s1)
```


DRIVERS

programmers can manipulate I/O devices directly by reading or writing the memory-mapped I/O registers as we have seen in the previous examples for i8085 and RISC-V.

However, it is better programming practice to call functions that access the memory-mapped I/O. These functions are called device drivers

Writing a device driver requires detailed knowledge about the I/O device hardware. Other programs call functions in the device driver to access the device without having to understand the low-level device hardware

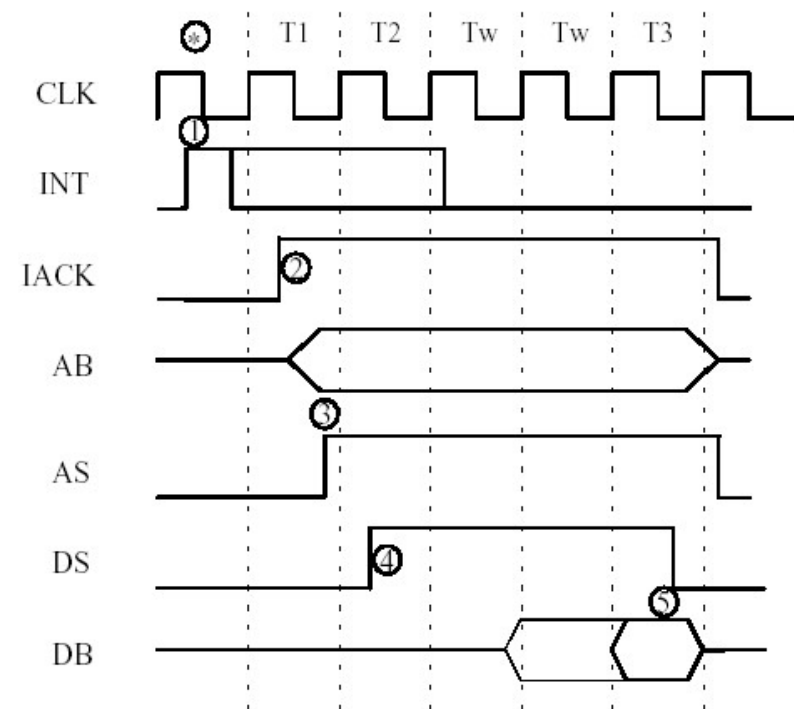
INTERRUPCIONS

La sincronització per consulta malgrat la seva senzillesa presenta el següent nombre d'inconvenients

- Si l'interval entre dues ocurrències és molt variable, l'atenció s'ha de programar per al pitjor cas i es perd molt de temps.
- La prioritat s'estableix per l'ordre de la consulta i és inamovible un cop programada
- En el cas de que un gran nombre de processos necessiti ser atès, o bé disposem de rutines diferents amb periodicitats adequades o bé s'ha de preveure el pitjor cas i es perd molt de temps.
- Si el procés és de curta durada però es produeix amb poca freq. l'atenció s'ha de preveure segons la durada i no segons la freq. i per tant es perd molt de temps inútilment

Cicles de la interrupció:

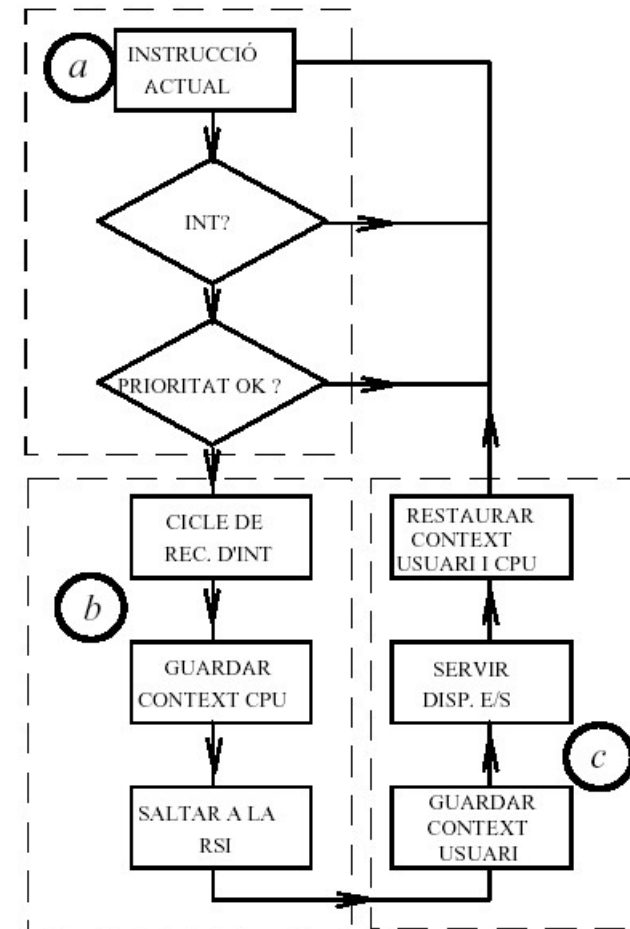
- 1.- Mostreig de la línia d'interruptió
- 2.- Establiment del cicle de reconeixement
- 3.- Establiment del codi específic de la interrupció que es reconeix
- 4.- Sollicitut de identificació de la interrupció
- 5.- Recepció de la identificació



* Últim cycle de rellotge de l'últim cycle màquina de la instrucció en curs

Rutina d'Atenció a la Interrupció (RAI)

- 1.- El uP executa una instr. i rep una interrupció
- 2.- La prioritat és superior al procés que està fent
□□ → S'aten la Int del disp. E/S
- 3.- Guarda PC, Registre d'estat a la pila
- 4.- Accedeix a la direcció de la RAI, carrega el nou PC
- 5.- Les interrupcions mascarables en aquesta fase **QUEDEN DESHABILITADES**
- 6.- L'usuari podrà habilitar algunes interrupcions dins de la RAI si ho creu convenient.
- 7.- S'haurà de guardar també el contingut d'alguns (o de tots) els registres utilitzats per l'usuari a la pila
- 8.- Execució de la part de codi que dona accés al dispositiu que ha generat la interrupció
- 9.- Recuperar el context d'usuari
- 10.- Retornar al programa principal



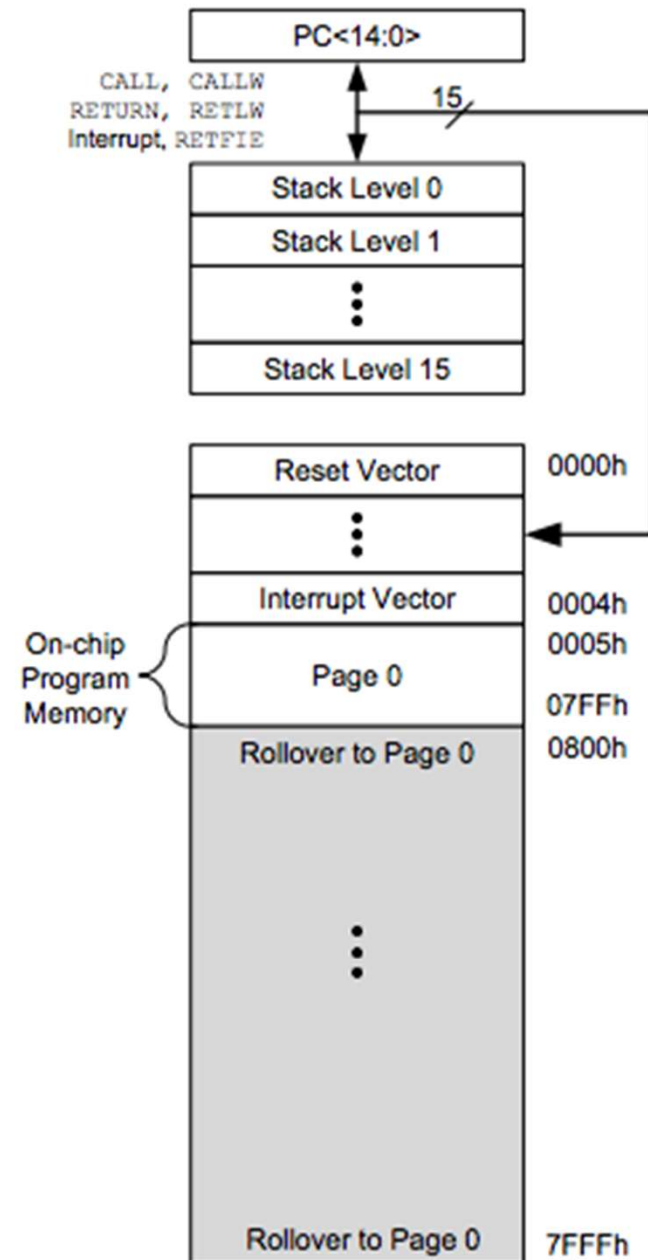
Interfícies d'entrada sortida

```

void main(void)
{
    T1CON = 0x01;           //Configure Timer1 interrupt
    PIE1bits.TMR1IE = 1;
    INTCONbits.PEIE = 1;
    RCONbits.IPEN=0x01;
    IPR1bits.TMR1IP=0x01;   // TMR1 high priority ,TMR1 Overflow Interr
    INTCONbits.GIE = 1;
    PIR1bits.TMR1IF = 0;
    T0CON=0x00;
    INTCONbits.T0IE = 1;    // Enable interrupt on TMR0 overflow
    INTCON2bits.TMR0IP=0x00;
    T0CONbits.TMR0ON = 1;
    while (1);
}

void interrupt tc_int(void)    // High priority interrupt
{
    if (TMR1IE && TMR1IF)
    {
        TMR1IF=0;
        ++tick_count;
        TRISC=1;
        LATCbits.LATC4 = 0x01;
    }
}

void interrupt low_priority    LowIsr(void)    //Low priority interrupt
{
    if(INTCONbits.T0IF && INTCONbits.T0IE) // If Timer flag is set & Interrupt
    {
        TMR0 -= 250;           // Reload the timer - 250uS per interrup
        INTCONbits.T0IF = 0;   // Clear the interrupt flag
        ADCON1=0x0F;
        TRISB=0x0CF;
        LATBbits.LATB5 = 0x01; // Toggle a bit
    }
    if (TMR1IE && TMR1IF)
    {
        TMR1IF=0;
        ++tick_count;
        TRISC=0;
        LATCbits.LATC3 = 0x01;
    }
}
    
```



El concepte de prioritat

Que passarà si s'activen dos interrupcions alhora??

El uP només pot executar una d'elles  Executarà la més
PRIORITARIA

Però, que s'entén per prioritat??

És com un semàfor, passa la interrupció que troba el semàfor en verd

El “semàfor” és el registre de màscara del uP

Tenim quatre alternatives per gestionar la prioritat. Cada una té avantatges i es poden fer servir depenent de les condicions

La gestió de la prioritat es pot fer de la següent forma:

- 1.- Nivells d'interrupció
- 2.- Consulta per software (software polling)
- 3.- Controlador de prioritat
- 4.- Prioritat segons la posició (Daisy-Chain)

Interfícies d'entrada sortida

NOM INT	INDEX PRIORITAT	DIRECCIÓ SALT	ACTIVACIÓ
TRAP	1	24 H	FLANC DESCENDENT I NIVELL ALT QUE S'HA DE MANTENIR FINS EL MOSTREIG
RST7.5	2	3CH	FLANC ASCENDENT
RST6.5	3	34H	NIVELL ALT FINS AL MOSTREIG
RST5.5	4	2CH	NIVELL ALT FINS AL MOSTREIG
INTR	5	Segons RSTn	NIVELL ALT FINS AL MOSTREIG

1.- Nivells d'interrupcions

Com a mínim qualsevol uP té 2 tipus d'interrupcions, es a dir, 2 nivells jeràrquics

- 1.- El més prioritari: No es pot emmascarar
es per això que es diu no-emmascarable
o NMI (de l'anglès *Non Maskable Interrupt*)
- 2.- El menys prioritari: És emmascarable i reb el nom genèric de
INT

Aquests tipus d'interrupcions tenen línies dedicades i sense codificar

No es necessita el nivell 3 d'establiment del codi de la interrupció ja que es troba implícit en la línia d'interrupció

3.- Controlador de prioritats

Mitjançant un hardware especialitzat que constitueix el controlador de prioritats, es fa que el codi d'identificació es correspongui amb el del dispositiu més prioritari.

Prioritat segons la posició (*Daisy-Chain*)

-Prioritats per *hardware* connectant els dispositius corresponents a aquest nivell en cadena, de tal manera que, amb la utilització d'una certa lògica, es faci que només un d'ells rebi servei per part del uP.

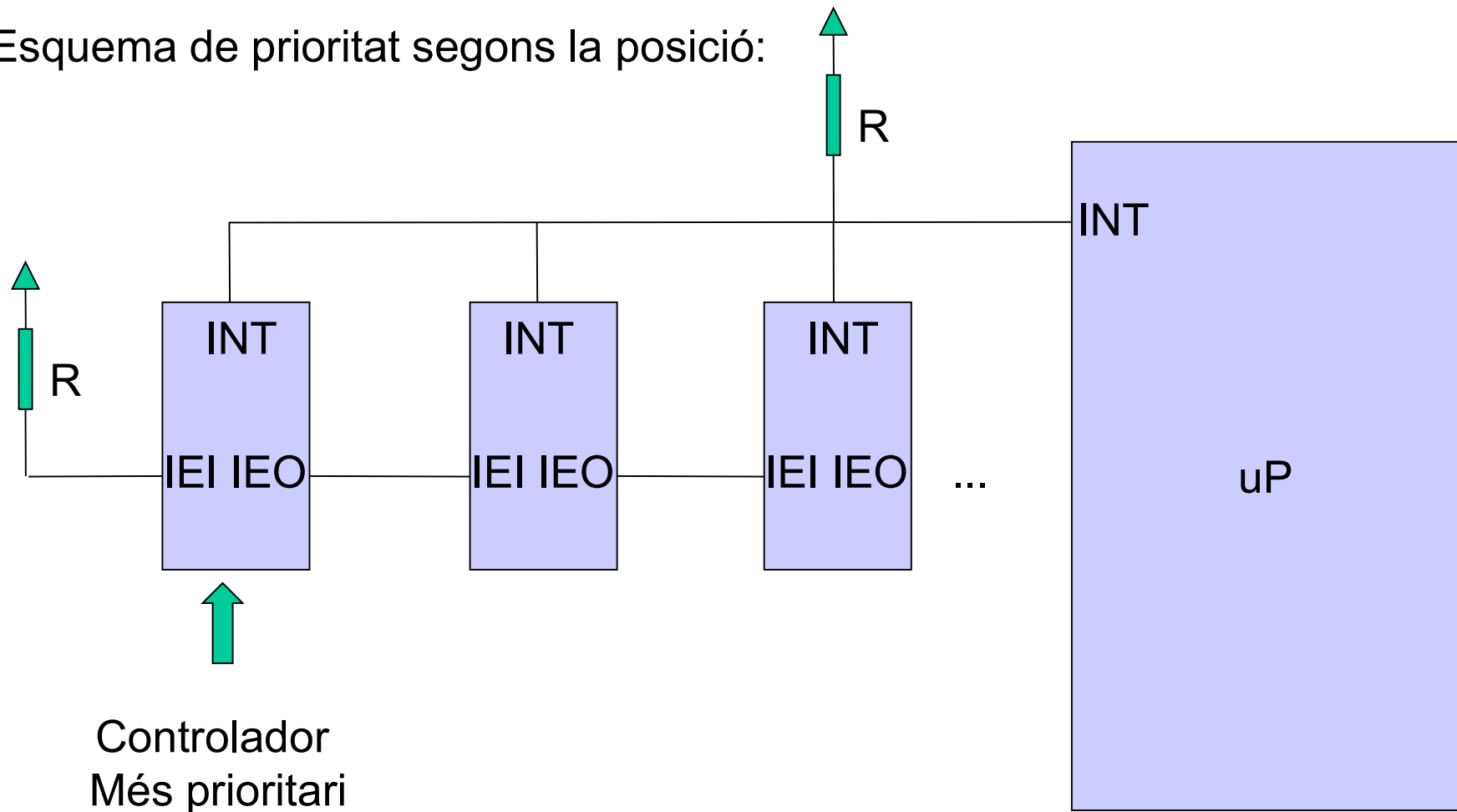
-Tenim dos habilitacions de les interrupcions:

IEI: Interrupt Enable Input

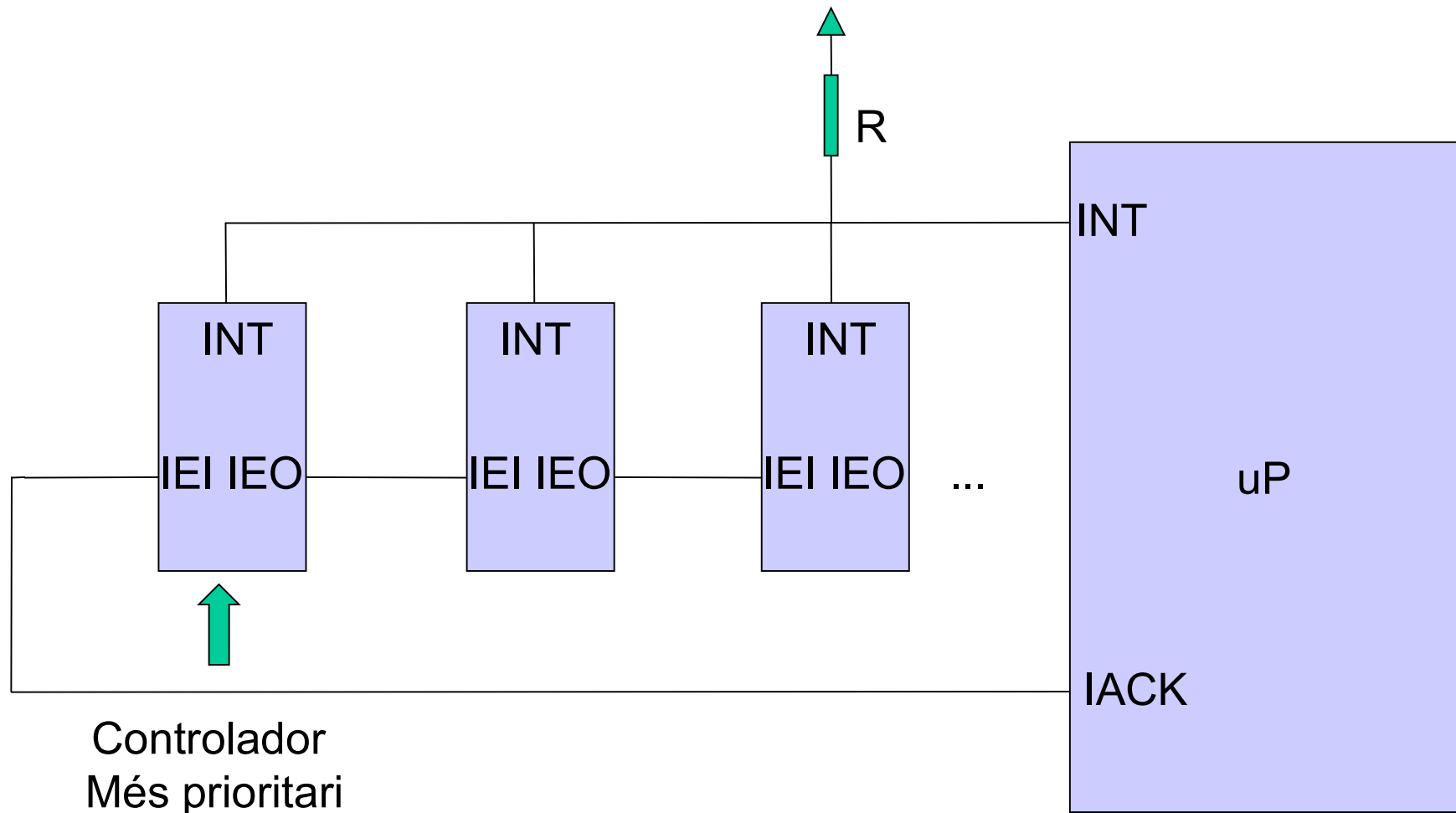
IEO: Interrupt Enable Output

- i la interrupció pròpiament dita: INT

Esquema de prioritat segons la posició:



En lloc de construir la cadena de prioritat sobre la petició de la interrupció, es pot construir sobre el reconeixement



Tipus d'interrupció en funció de la seva disponibilitat

-Interrupcions directes

-Interrupcions autovectoritzades

-Interrupcions vectoritzades

NOM INT	INDEX PRIORITAT	DIRECCIÓ SALT	ACTIVACIÓ
TRAP	1	24 H	FLANC DESCENDENT I NIVELL ALT QUE S'HA DE MANTENIR FINS EL MOSTREIG
RST7.5	2	3CH	FLANC ASCENDENT
RST6.5	3	34H	NIVELL ALT FINS AL MOSTREIG
RST5.5	4	2CH	NIVELL ALT FINS AL MOSTREIG
INTR	5	Segons RSTn	NIVELL ALT FINS AL MOSTREIG

FORMATO	RST	CONTADOR DE PROGRAMA
1100 0111	C7	0000 0000 0000 0000 0000H
1100 1111	CF	0000 0000 0000 1000 0008H
1101 0111	D7	0000 0000 0001 0000 0010H
1101 1111	DF	0000 0000 0001 1000 0018H
1110 0111	E7	0000 0000 0010 0000 0020H
1110 1111	EF	0000 0000 0010 1000 0028H
1111 0111	F7	0000 0000 0011 0000 0030H
1111 1111	FF	0000 0000 0011 1000 0038H

Interrupcions directes

L'adreça de la RSI es troba fixada per la pròpia estructura del uP
(*interrupció directa*)...

Inconvenient:

La seva rigidesa...

Avantatges:

Molt senzill

No necessita de la intervenció del dispositiu E/S  Reconeixement curt

Interrupcions autovectoritzades

Fa servir un vector d'interrupció per defecte programat per l'usuari

La RAI pot situar-se a qualsevol lloc

No cal subministrar aquest vector durant el reconeixement de la interrupció

Avantatges:

Més flexible que el tipus d'interrupcions directes

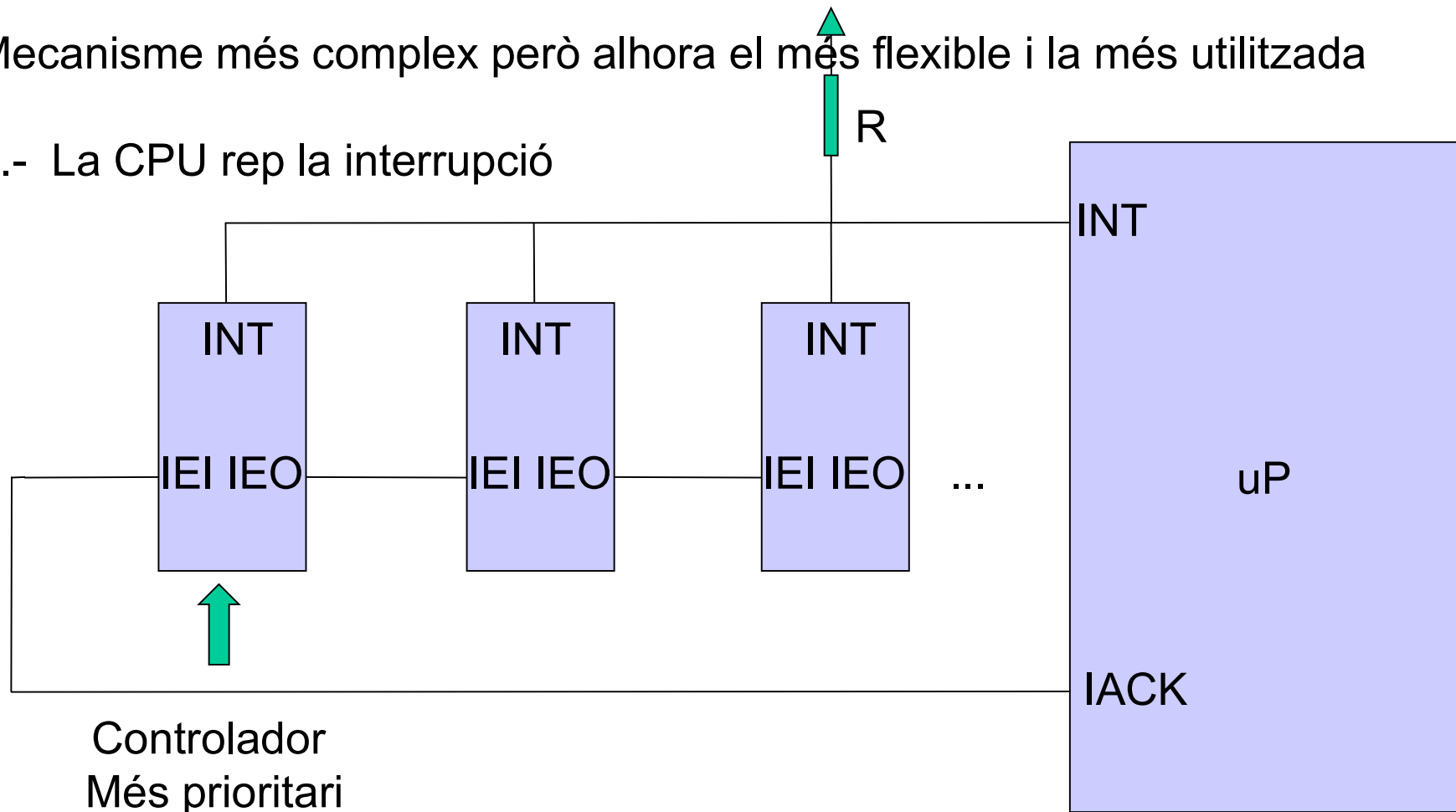
Inconvenients:

Requereix una certa participació dels dispositiu E/S

Interrupcions vectoritzades

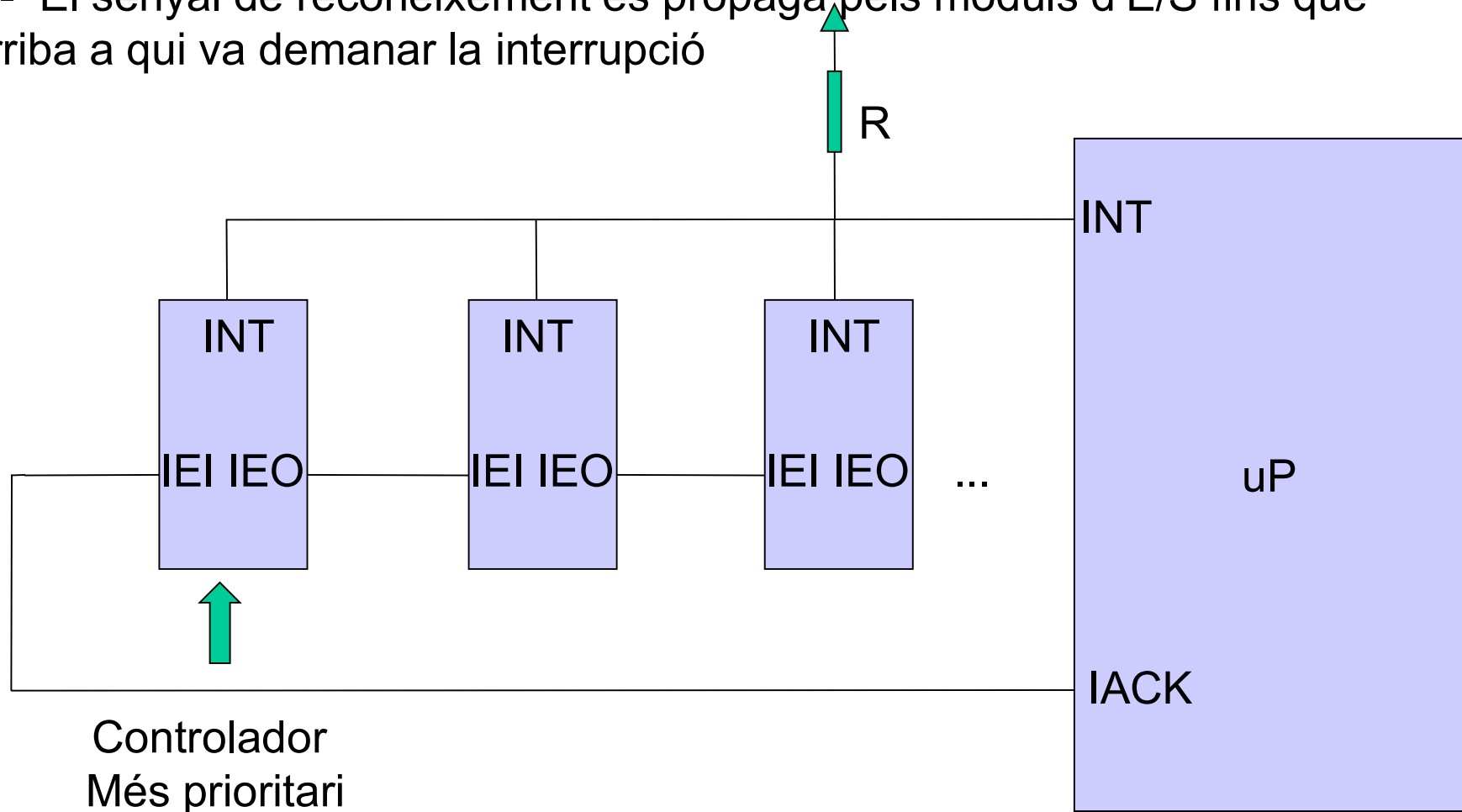
Mecanisme més complex però alhora el més flexible i la més utilitzada

1.- La CPU rep la interrupció

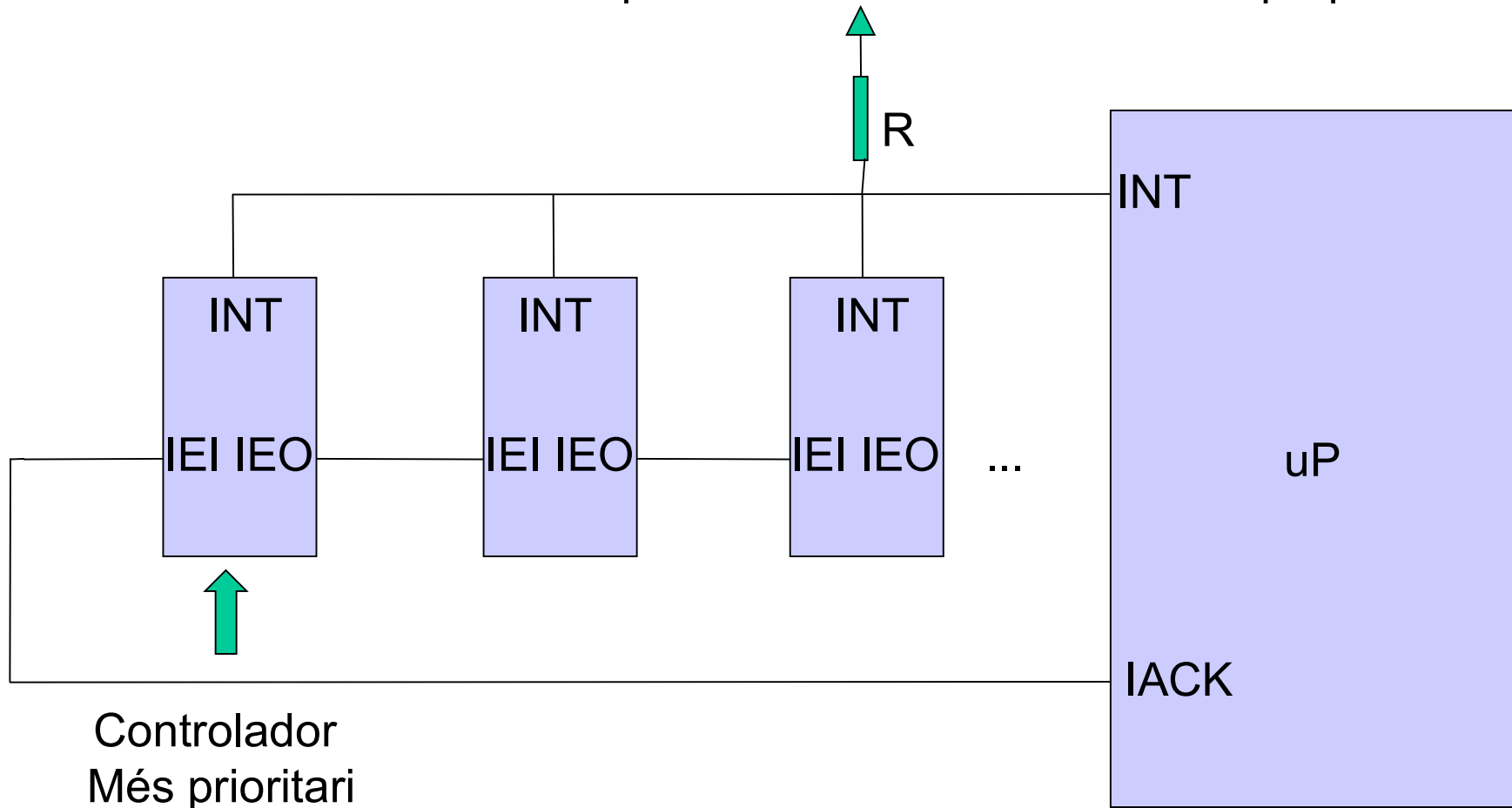


2.- Activació reconeixement de la interrupció

3.- El senyal de reconeixement es propaga pels mòduls d'E/S fins que arriba a qui va demanar la interrupció



- 4.- El mòdul respon col·locant una paraula (vector) al bus de dades
- 5.- El vector és la direcció del mòdul E/S o algun identificador específic
- 6.- La CPU utilitza el vector com apuntador a la rutina de servei apropiada



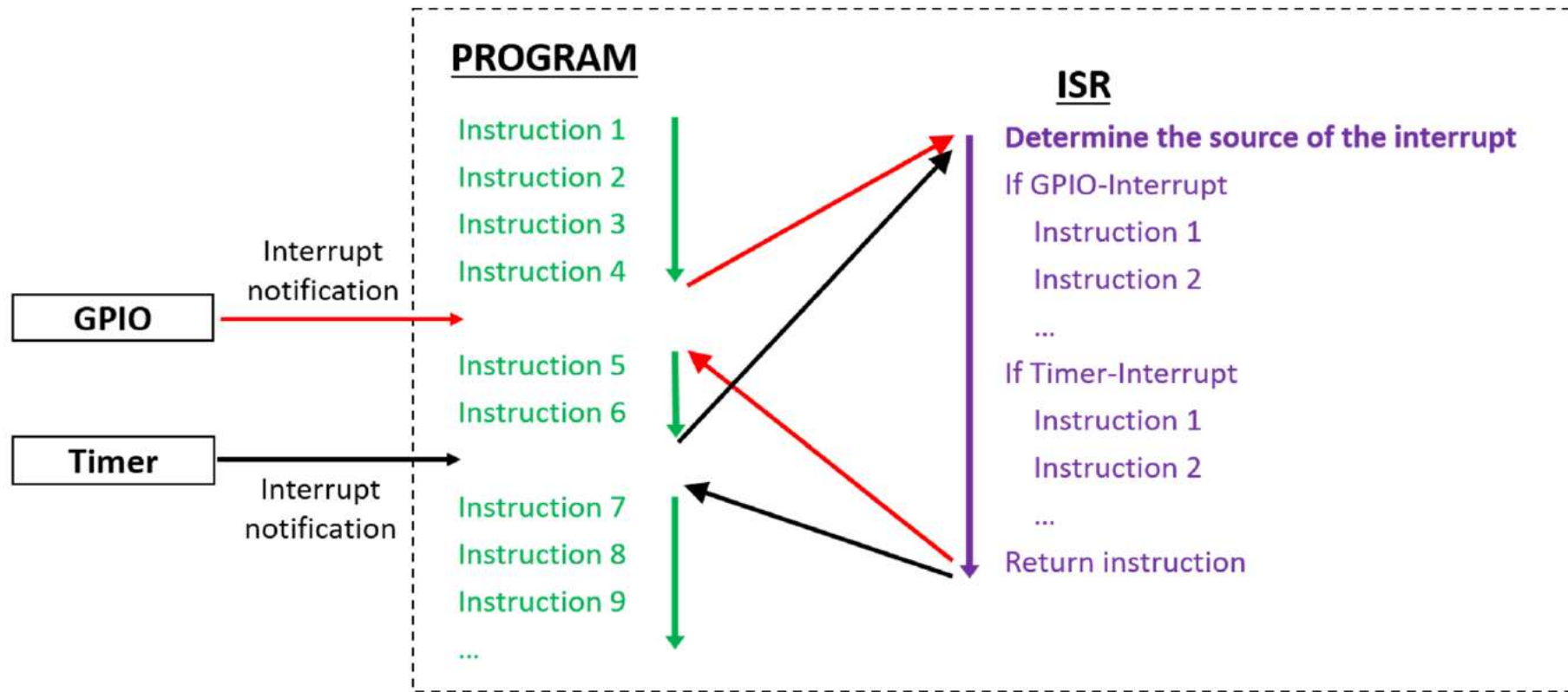


Figura 1. Exemple amb dos interrupcions en mode d' un sol vector

Interfícies d'entrada sortida

```
/* INITIALIZE INTERRUPT LINE IRQ4 */  
ExternalIntLine_Initialization(4, 6, GPIO_ISR); /* Initialize line IRQ4 with a priority of 6. Set GPIO_ISR as the Interrupt Service Routine */  
ExternalIntLine_Initialization(3, 6, PTC_ISR); /* Initialize line IRQ3 with a priority of 6. Set PTC_ISR as the Interrupt Service Routine */
```

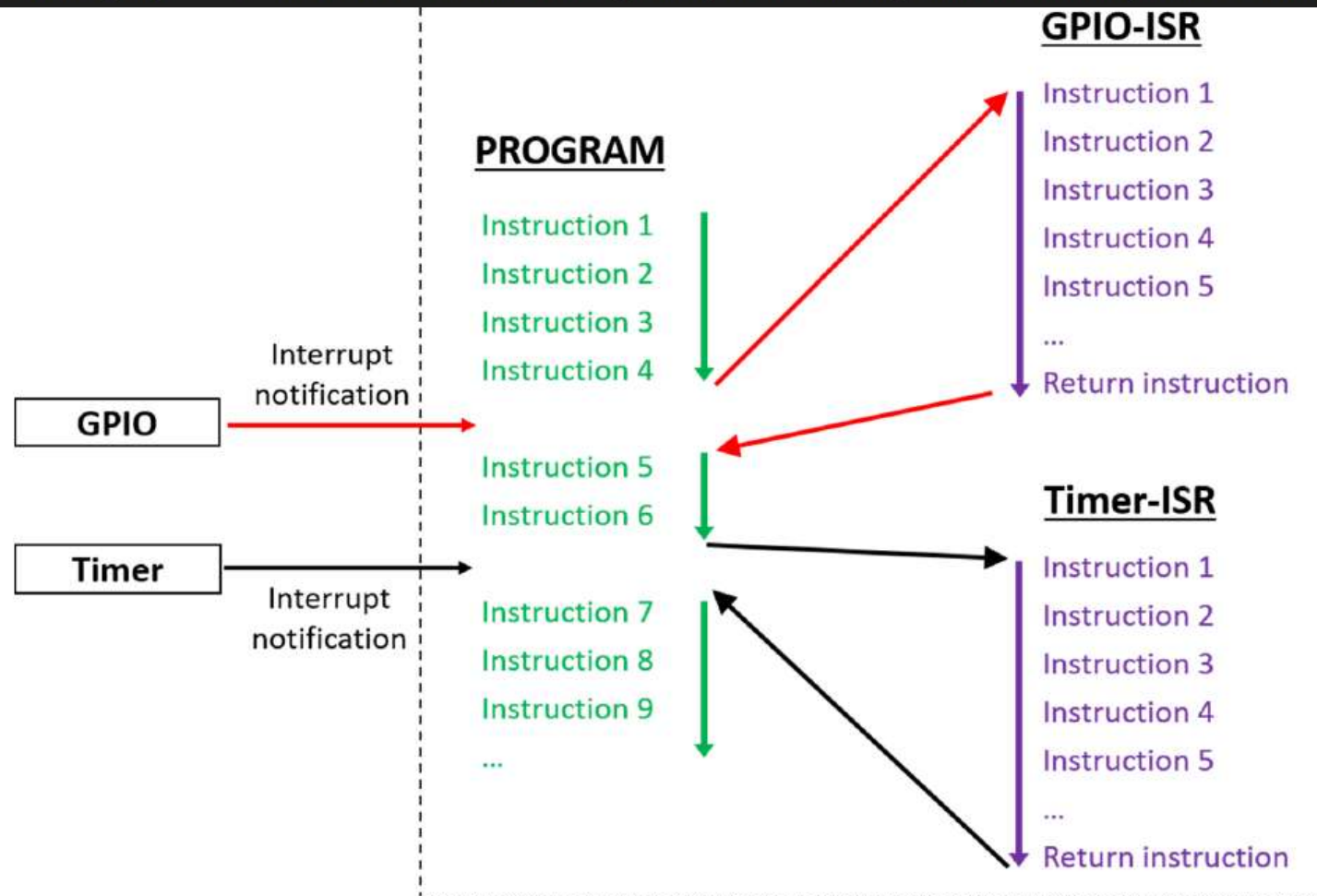
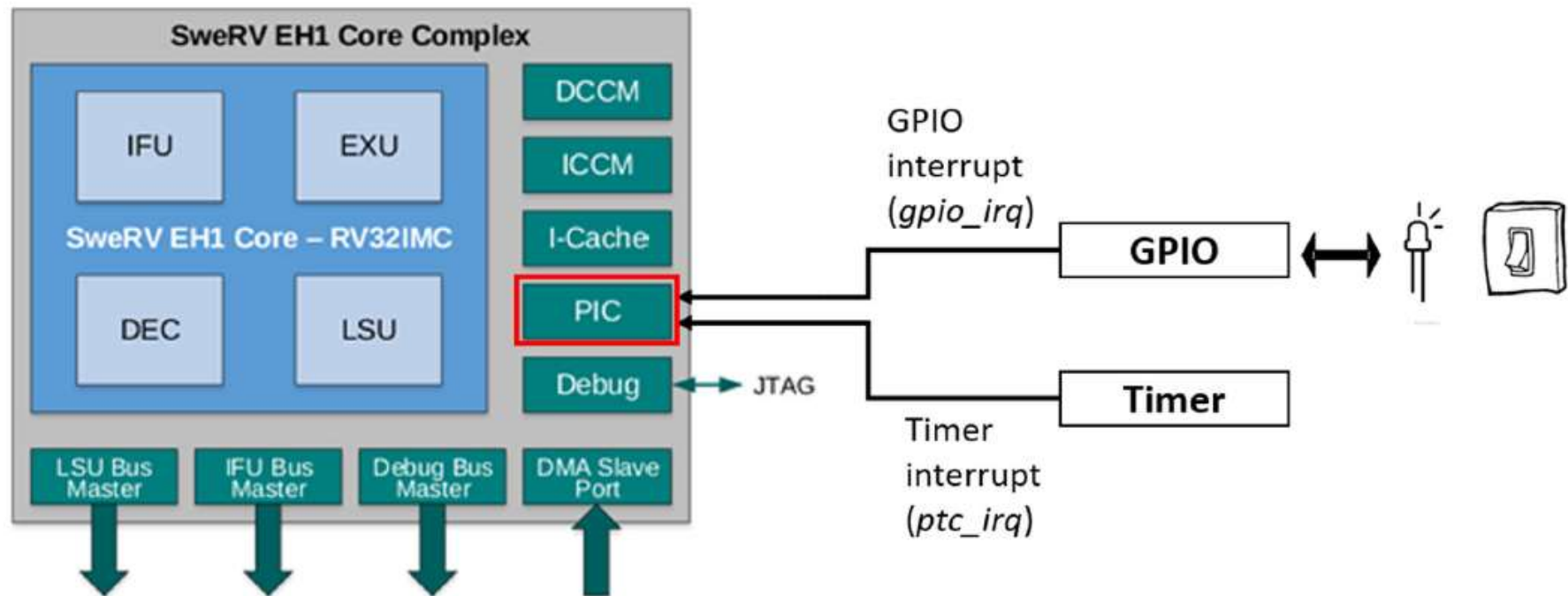


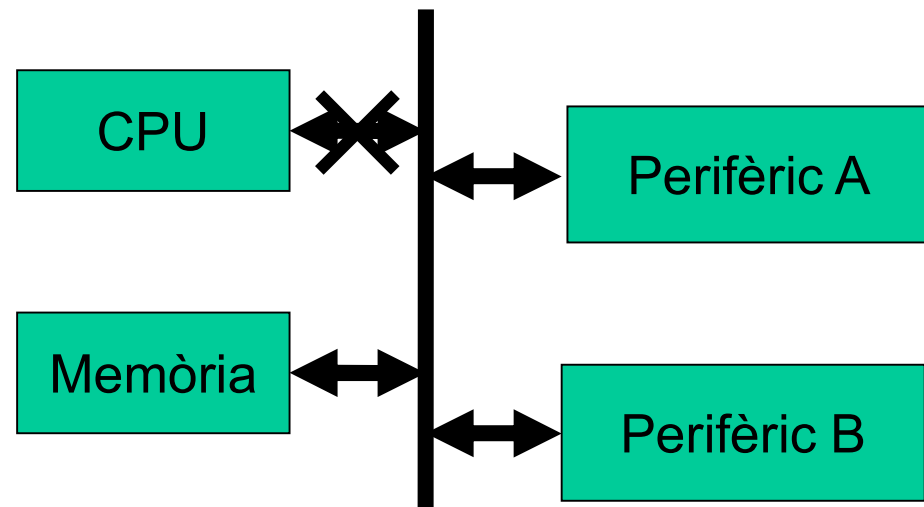
Figura 218. Exemple de dos interrupcions en mode multi-vector



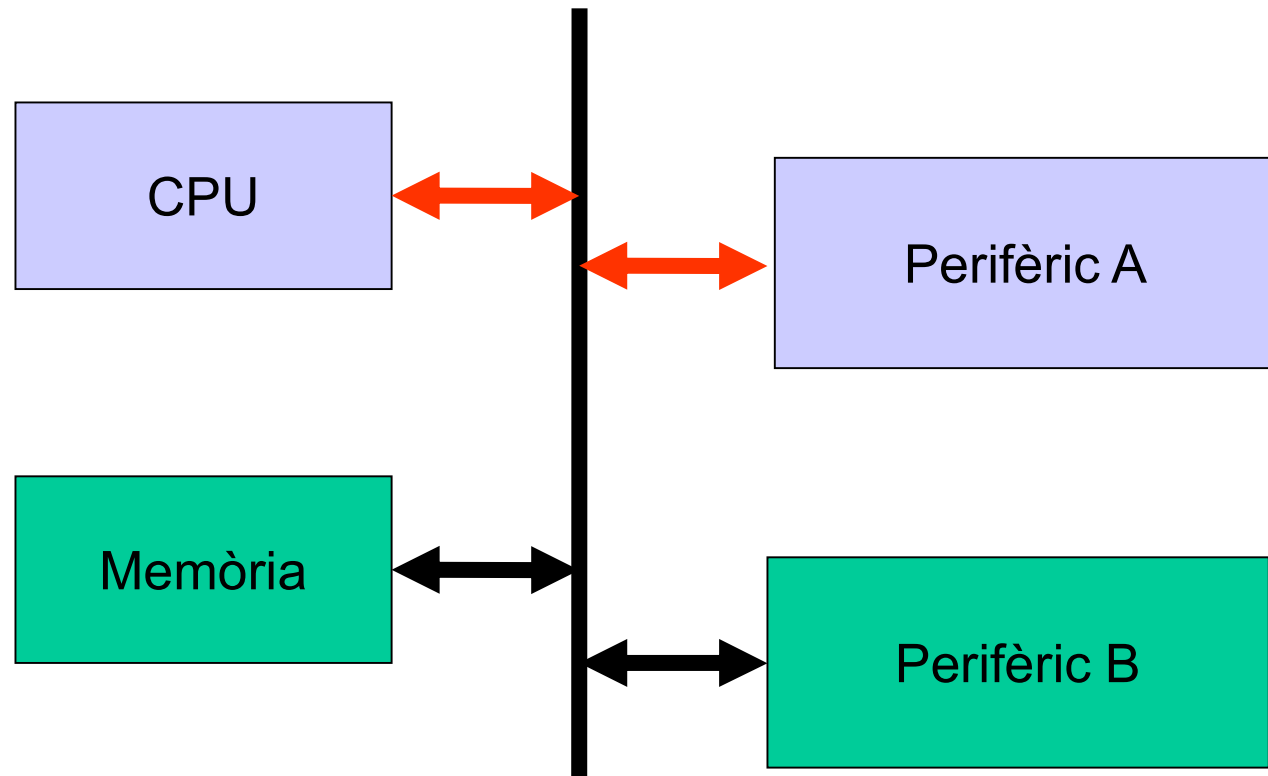
DIRECT MEMORY ACCES (DMA)

- Les transferències entre memòria i perifèric estan controlades per la interfície o dispositiu E/S
- Desconnecta del BUS a la CPU, afegint un sistema digital especialitzat en fer les transferències el més ràpid possible.
- El sistema digital es coneix amb el nom de Controlador de DMA o DMAC
- Crea de forma automàtica tots els senyals necessaris per fer les transferències de la forma més ràpida possible.
- ES EL SISTEMA DE GESTIÓ DE TRANSFERENCIA MES RÀPID

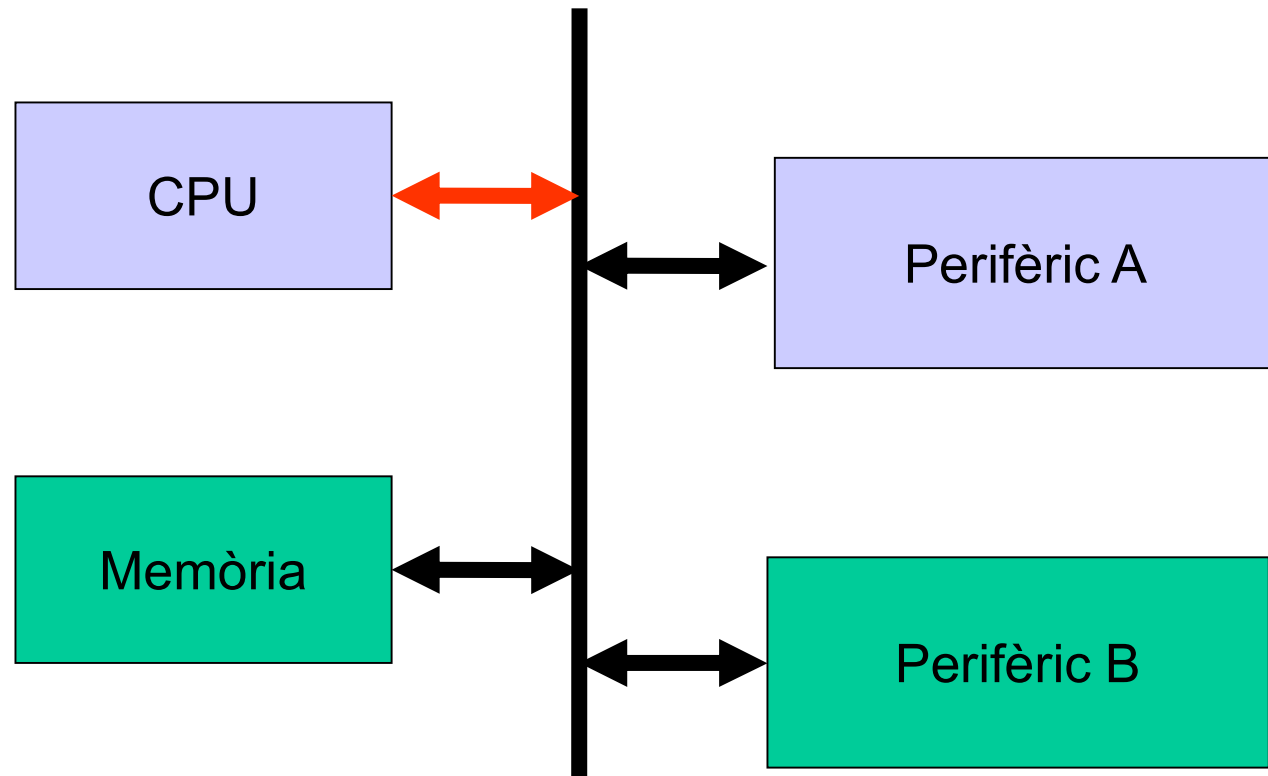
- L'accés a la memòria es fa de forma seqüencial.
- No necessita per tant perdre temps buscant l'ordre d'accés a la memòria
- Aquesta topologia permet fer transferències entre perifèric i perifèric
- Aquestes transferències acostumen a ser en paral·lel.



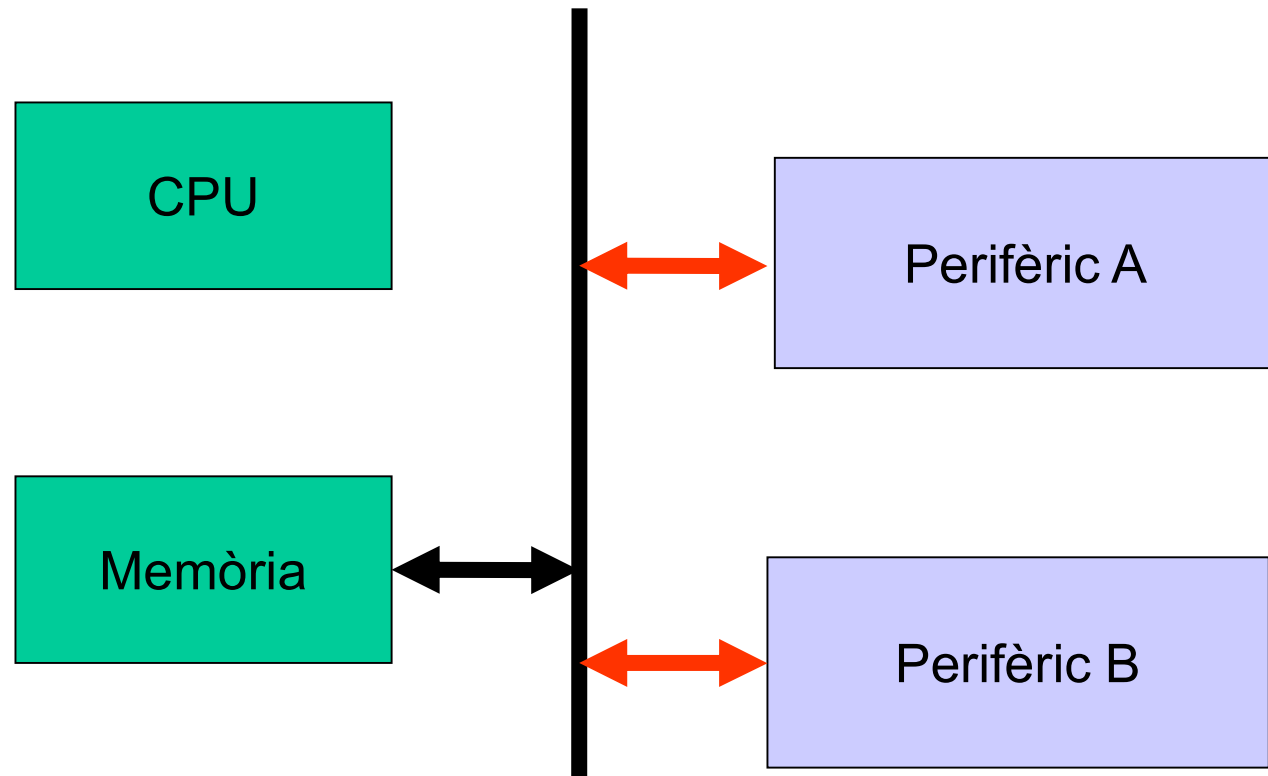
- 1.- El perifèric demana a la CPU el control del BUS (comunicació a partir del bus de control)



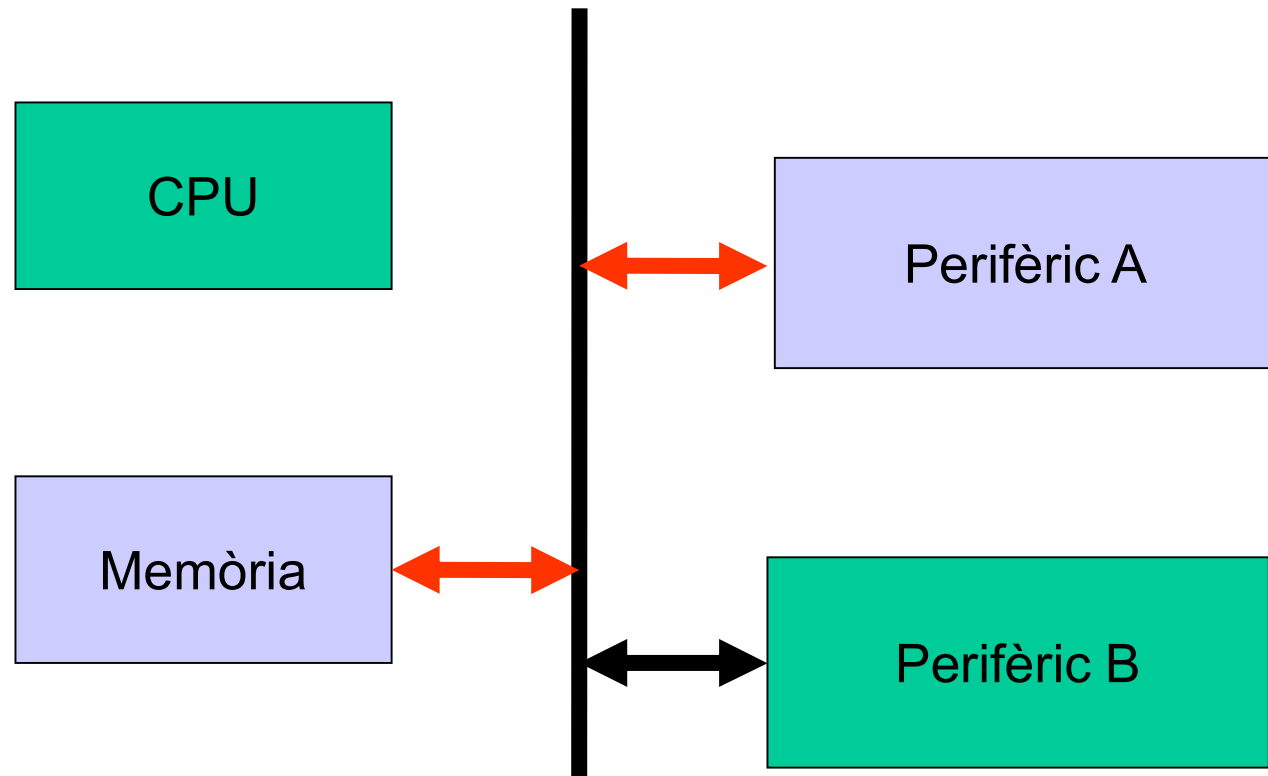
2.- La CPU dona el control del BUS a la interfície i es retira del BUS



3.- S'estableix la comunicació directa Perifèric perifèric



...o perifèric memòria



4.- Finalment, es retorna el control del BUS a la CPU

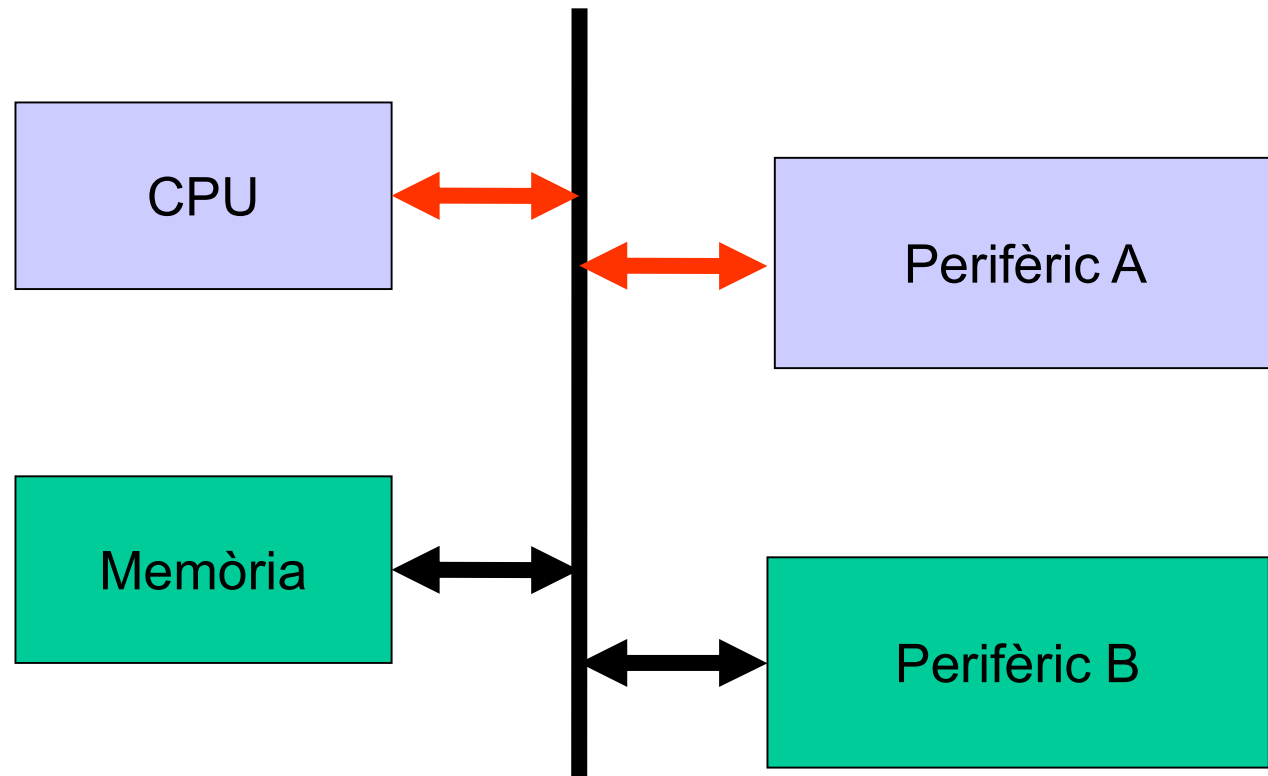
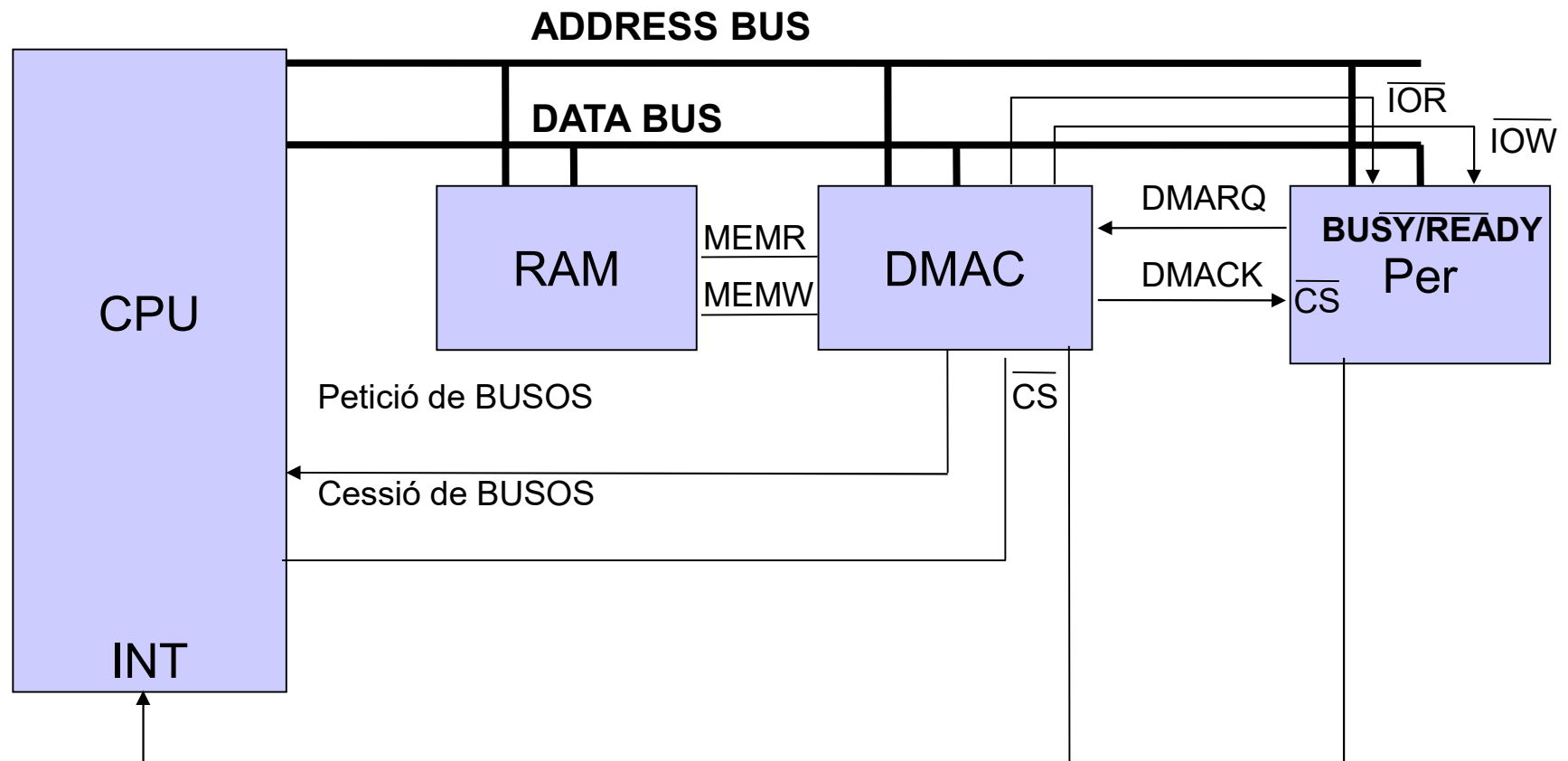


Diagrama de blocs d'una transferència controlada per DMAC




El DMAC s'ha d'inicialitzar al començar el procés perquè pugui fer les transferències.

¿Qui fa aquesta inicialització?

... La CPU dient-li a la DMAC

- Entre que punts es farà la transferència
- L'adreça inicial i la grandària del bloc que s'ha de transferir i si les adreces de les posicions de memòria que processem amb el DMA augmenten o disminueixen
- El mode com es produeix la transferència

Un cop la interfície té els busos  el DMAC va generant amb un comptador intern les adreces de les posicions de memòria i la selecció dels perifèrics que participen en la transferència

El DMAC disposa a la seva estructura de 4 registres (com a mínim...):

1.- Registre d'adreces: Conté l'adreça de memòria on s'ha de transferir la següent paraula, incrementant-se automàticament després de cada transferència.

2.- Registre d'adreçament de perifèrics

3.-Comptador de paraules: Conté el nombre de words a transferir. Es decrementa cada cop que es completa una transferència. Quant arriba a 0, genera un senyal de finalització de l'operació d'E/S

4.-Registre de dades: Conté la paraula a transferir en cada operació elemental.

Modes de transferència treballant amb DMAC:

- 1.- Mode byte: Enviem un únic byte un cop tenim els busos. Quan el perifèric està OK tornem els busos
- 2.- Mode burst o demanda: DMAC transfereix les dades fins que el perifèric diu prou. La transferència es fa per trossos de missatge (burst)
- 3.- Mode bloc: Anomenat també continu. El bus no es deixa fins que s'ha transferit el bloc sencer

- Un cop finalitzada la transferència es demana una interrupció que serà atesa un cop la CPU tingui els busos.
- La petició d'interrupció serveix per indicar-li a la CPU que s'ha terminat el procés de DMA
- En les transferències perifèric-memòria i perifèric-perifèric és el perifèric qui demana DMA donant l'ordre al DMAC.
- En les transferències memòria-perifèric i memòria-memòria és la CPU qui comença el procés DMA donant l'ordre al DMAC
- Els intercanvis en un procés DMA poden fer-se passant per la CPU o directament, es coneixen com *transferència seqüencial* i *transferència simultània*:

La transferència seqüencial activa en seqüència els senyals d'accés a memòria o dispositiu E/S, MEMR o MEMW i IOR o IOW

La transferència simultània activa alhora l'ordre de lectura o escriptura a la memòria i al perifèric