ARQUITECTURA DE COMPUTADORS I PERIFÈRICS

SOLUCIONS PROBLEMES

Entrada/Sortida

ARQUITECTURA DE COMPUTADORS I PERIFÈRICS

Escola d'Enginyeria **PROBLEMES**

Enginyeria Telecomunicacions **ENTRADA/SORTIDA**

Universitat Autònoma de Barcelona

Es té un sistema computador amb un únic bus compartit que gestiona, entre d'altres, una pantalla alfanumèrica (per mostrar resultats) i un disc dur (per emmagatzemar dades). Per controlar aquests dispositius, aquest sistema inclou els següents components:

- Una CPU amb el mateix repertori d'instruccions vist als problemes del tema 1 del processador, que executa 200 MIPS (milions d'instruccions per segon) i amb un sistema d'interrupcions vectoritzades. Els **registres d'aquesta CPU son de 32 bits**, però permet carregar-li'ls dades de **8** i **16** bits (**halfword**) des de la memòria, situant **0** en els **bits més alts (o significatius)** del registre (del 31 al 8, per les 8 bits i del 31 al 16 per les de 16 bits). Per operar amb aquestes dades , s'han afegit les següents instruccions:

  - byte: Implica la lectura/escriptura de la dada (1 byte) de 1 adreça de memòria :

  LB Rd, desplaçament(Rm): Llegir una dada (tipus byte) de la memòria.

  $Rd_{(8..0)}$<- [mem (Rm + desplaçament)]

  SB Rd, desplaçament(Rm) Escriure una dada (tipus byte) a memòria.

  $Rd_{(8..0)}$ -> [mem (Rm + desplaçament)]

  - halfword (16 bits): Implica la lectura de les dades (d'un byte cadascuna) de 2 adreces de memòria consecutives.

  LH Rd, desplaçament(Rm): Llegir una dada (tipus hlafword) de la memòria.

  $Rd_{(15..8)}$<- [mem (Rm + desplaçament+1)] i *$Rd_{(7..0)}$<- [mem (Rm + desplaçament)]*

  SH Rd, desplaçament(Rm) Escriure una dada (tipus hlafword) a memòria.

  $Rd_{(15..8)}$ -> [mem (Rm + desplaçament+1)] i *$Rd_{(7..0)}$ -> [mem (Rm + desplaçament)]*

  Exemples:

  Si **R3** conté **0x0000F100** (es una adreça de memòria)

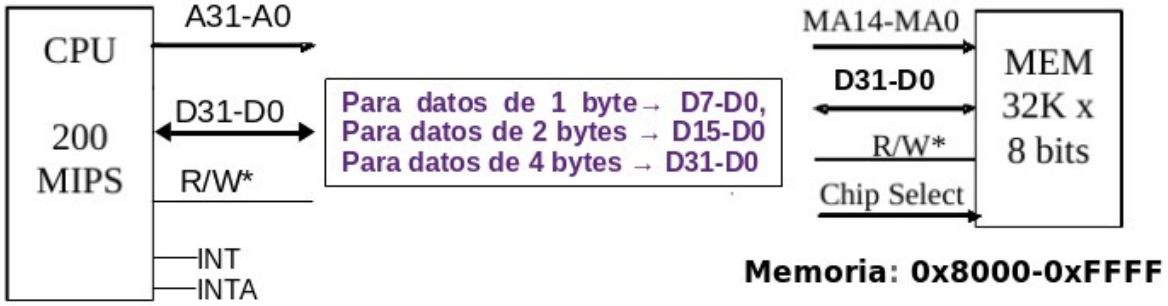
  La instrucció **LB R1, 0(R3)** situa el contingut de l'adreça de memòria (**0x0000F100 + 0)** (1 byte) en els **8 bits de la part baixa de registre R1** (i 0 als 24 bits de la part alta).

  La instrucció **SH R2, 4(R3)** situa els 16 bits de la part baixa de R2 al contingut de les adreces de memòria (**0x0000F100+4) i** (**0x0000F100+4+1).** Això es:

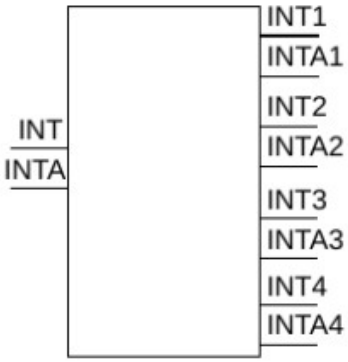
  $R2_{(15,..,8)}$ -> [0x0000F105] i $R2_{(7,..,0)}$ -> [0x0000F104]

- **El espai d'adreces d' E/S** del sistema **NO està separat del de memòria (es memory mapped).**
- Un mòdul de memòria de 32K x 8bits
- Un mòdul **controlador I2C** per dispositius connectats a través d'un bus I2C (serie síncron) amb capacitat d'interrupció al qual està connectat la pantalla alfanumèrica. El seu **identificador d'interrupció es el 11**.
- Un mòdul **controlador d'Interrupcions** amb capacitat per a gestionar fins a 4 dispositius i de generar els **vectors d'interrupció de 8 bits** corresponents a cada dispositiu.
- Un mòdul **controlador de disc** amb capacitat d'interrupció i capaç de comunicar-se amb un controlador de DMA. La taxa de transferència del disc és de 200 MB/s. El seu **identificador d'interrupció es el 13**.
- Un mòdul **controlador de DMA**. El seu **identificador d'interrupció es el 15**.
- La gestió de les interrupcions **es apropiativa**, això es, els sistema d'interrupcions **no queda inhibit** mentre s'atén a una interrupció.

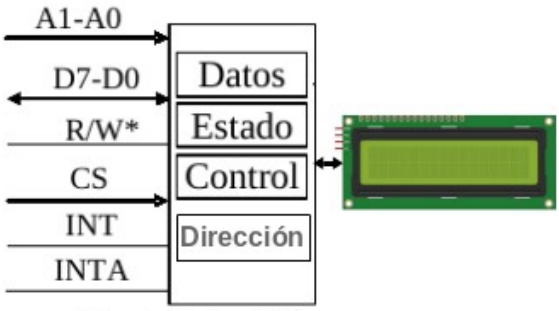
La figura de la pàgina següent indica les adreces en les que es pot accedir a cadascun del ports d' E/S dels controladors.



Memoria: 0x8000-0xFFFF

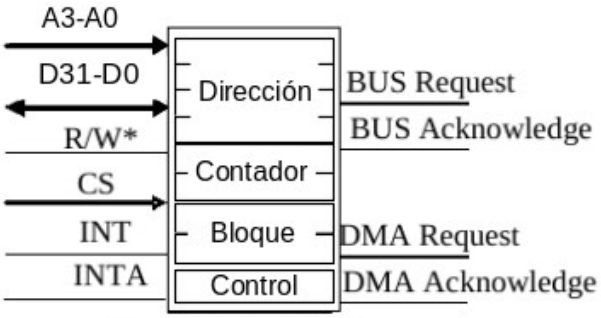


Controlador Interrupciones



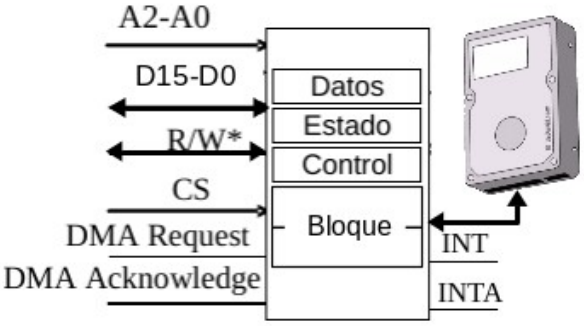
Controlador I2C

Datos : 0x200
Estado: 0x201
Control: 0x202
Dirección: 0x203



Controlador DMA

Dirección: 0x300-0x303
Contador: 0x304-0x305
Bloque: 0x306-0x307
Control: 0x308



Controlador Disco

Datos : 0x208
Estado: 0x209
Control: 0x20A
Bloque: 0x20B-0x20C

TASA TRANSFERENCIA: 200 MB/s

Registres de la CPU relacionats amb la gestió d'interrupcions:

Apart dels registres vistos de la CPU en el tema 1, aquesta també té els següents **registres de control i estat (CSR)** per la gestió de les interrupcions (i les excepcions):

- ***Registre d'estat de la màquina (MSTATUS):*** El bit 3 d'aquest registre **habilita i deshabilita globalment les interrupcions.** Si val 1 les interrupcions estan habilitades i la CPU les acceptarà, si val 0, estan deshabilitades i la CPU no acceptarà cap interrupció.
- ***Registre MIE:*** **Habilita o deshabilita una interrupció determinada (habilitació local d'interrupcions).** si el **bit i de MIE** està a **1**, la interrupció associada a aquest bit està **habilitada** i la CPU l' acceptarà (sempre que les interrupcions estiguin habilitades globalment). Si **val 0** està **deshabilitada** i la CPU no l'acceptarà. Els bit del MIE associat a la interrupció del **controlador I2C** es el **11**, el del **controlador de disc** es el **13** i el del **DMA** es el **15.**
- ***Registre MTVEC:*** Conte l'adreça base (a on comença) de la **taula del vector d'interrupcions.** En aquest sistema la taula de vectors d'interrupcions està situada a les primeres posicions de la memòria principal, per tant el valor d'aquest registre serà **0x8000**
- ***Registre MCAUSE:*** Conte el identificador de la font d'interrupció. Per exemple, si ha interromput el **controlador I2C**, aquest registre tindrà el valor de **11.**

IMPORTANT: Mentre no s'indiqui el contrari, es considerarà que les interrupcions estan habilitades, tant globalment com localment.

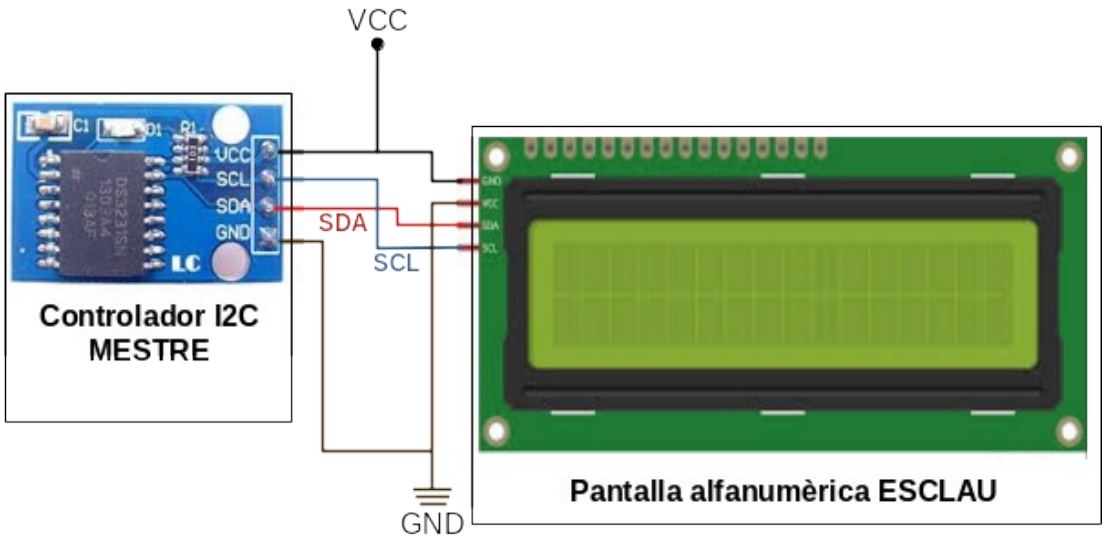
Controlador I2C

El bus I2C, que utilitzen per intercanviar informació el controlador I2C i la pantalla alfanumèrica, es un bus de comunicacions **serie síncron** .
Les característiques més importants d'aquest bus son:

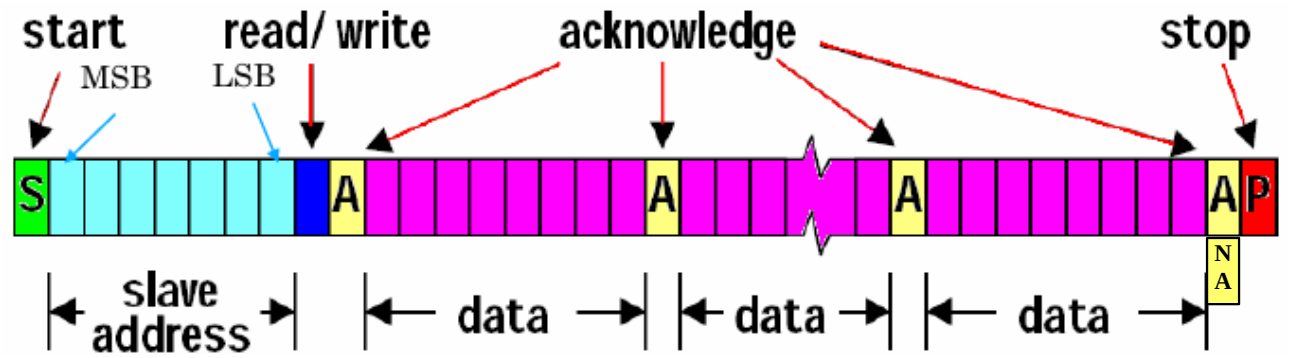
- Només necessita dues línies per transmetre dades, la del rellotge (**SCL**) i la de dades (**SDA**).
- Tots els dispositius connectats a aquest bus poden ser **mestres** o **esclaus**, però sempre es el dispositiu **mestre** (en el nostre cas es el controlador I2C) el que inicia una transmissió.
- Independentment que el dispositiu sigui mestre o esclau, pot enviar o rebre informació (es el mestre qui decideix si vol rebre o enviar informació a l'esclau).
- Cada dispositiu es reconegut amb **adreça única de 7 bits** (el **controlador I2C te la 0x28** i **pantalla te la 0x2C**).
- Les dades es transmeten en blocs de 8 bits i darrere de cada un d'aquest blocs, s'ha de rebre una senyal de reconeixement (handshake)
- Velocitat de transmissió: 3,4 Mbits /s

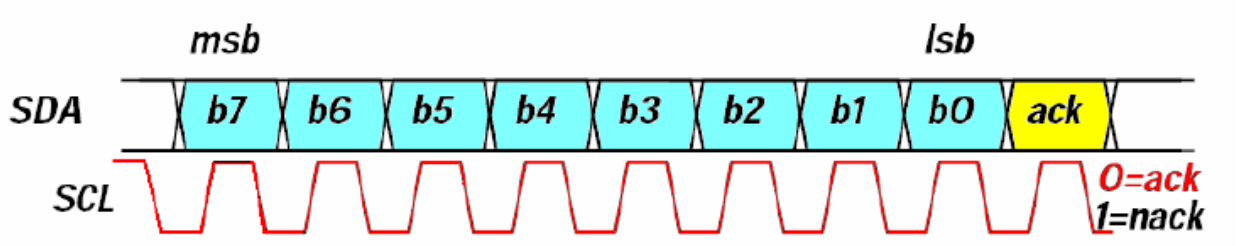
Esquema de connexió i format d'un missatge

L'esquema de connexió entre el controlador I2C i la pantalla alfanumèrica:



El format dels missatges enviats pel bus de dades (**SDA**) i la seva sincronització amb la senyal de rellotge (SCL) es les següents:





msb: Bit més significatiu (Most significant bit)
lsb: Bit menys significatiu (low significant bit)

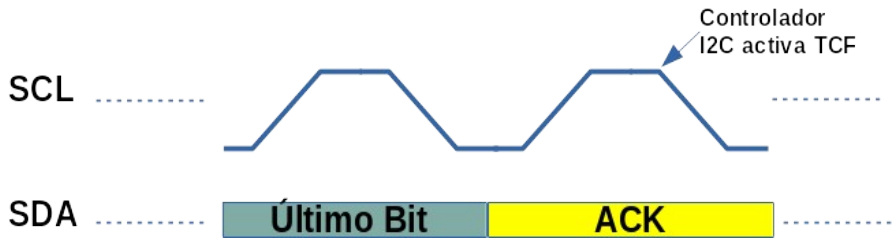
Transmissió d'un missatge

La transmissió d'un missatge pel bus **SDA** segueix els següents principal passos:
1) Sempre es el dispositiu **Mestre** que inicia la comunicació i això ho fa posant la senyal de **START** i el **primer byte del missatge**, en el qual situa l'**adreça del dispositiu Esclau (bits 7 al 1)** i al **bit 0** si es una operació de **lectura (Esclau -> Mestre)** o **escriptura (Mestre -> Esclau).**
2) Durant el següent cicle del rellotge del **SLC,** el dispositiu **Esclau** envia un **Acknowledge (ACK, té un valor de 0)** per confirmar que ha rebut la petició del **Mestre.**
3) A continuació es transmeten **totes les dades de 1 byte (1 bit per cicle del SCL)** fins l'última. Si es una **lectura,** el **Mestre situa les senyals de ACK** després de cada byte per confirmar que ha rebut les dades. Si es un **escriptura** es l'**Esclau que situa les senyal de ACK** després de cada dada per confirmar la recepció de les dades. En cas d'una **lectura**, després de **l'ultim byte de dades**, el **Mestre envia**, una senyal de **NACK (te el valor de 1)** enlloc d'un **ACK.** Per informar que el missatge ha finalitzat, el **Mestre envia, després de l'últim ACK** o **NACK,** la senyal de **STOP** al **Esclau.**

Registres d'Entrada/Sortida del controlador del I2C:

- **Dades:** Conte el byte a enviar o rebut.
- **Control:** Controla el funcionament del controlador I2C, els seus bits son (el reste son indiferents):
  - **7 (IICEN):** Activa el el controlador (1: actiu, 0: inactiu)
  - **6 (IICIE):** Activa la generació d'interrupcions (1:actiu, 0:Inactiu)
  - **5 (MST):** Selecciona si actua com a Mestre o Esclau (1:Mestre, 0:Esclau)
  - **4 (TX**): Selecciona si el Mestre envia o rep dades (1:Envia, 0:Reb)
  - **3 (TXAC):** Activa la generació de senyals de reconeixement (1:Actiu, 0:inactiu)
- **Estat:** Conte d'informació sobre el funcionament del controlador, els seus bits son (el reste son indiferents):

- **7 (TCF):** Informa si s'ha completat un transferència (1: Completada, 0: No completada). En cas de lectura indica que ja hi es el byte al registre de dades i si es un escriptura, el registre de dades ja està preparat per rebre la dada a enviar. Aquest **bit s'esborra al fer una lectura o escriptura del registre de dades**. Perquè es pugui comprovar si la senyal ACK ha estat col·locada per l'esclau, aquest flag s'activa al pols de baixada del cicle del SCL corresponent al bit d'aquesta senyal ACK (transmesa pel bus SDA).



- **2 (SRW):** Direcció de transmissió indicada pel Mestre a l'Esclau (1:Envia, 0:Reb)
- **0 (RXAC):** Recepció de les senyals de reconeixement (1:Rebut **ACK** , 0:No Rebut **ACK**)

• **Adreça:** Conté l'adreça de 7 bits (bits 7 al 1) del dispositiu en mode **Esclau**.

Controlador disc dur

Els dispositius d'emmagatzematge de dades, com els discs durs, DVD's, memòries USB, etc, utilitzen diferents **formats físics i protocols** (suport magnètic, òptic, interfície SATA, M.2, etc) per a emmagatzemar les dades dels diferents fitxers del sistema (executables, documents, llibreries, imatges, etc). Per a aïllar al programador (i al microprocessador) dels diferents formats físics dels dispositius d'emmagatzematge, aquests són definits com uns **dispositius d'emmagatzematge lògic,** els quals estan formats per una **llista de N blocs** (bloc 0, bloc 1,...,bloc N-1, per aquest controlador N es 2^{16}) **d'una mida de M bytes cadascun,** pels exercicis d'aquest tema, **M** és **1 Kbyte.** Per tant, pel programador cada fitxer estarà emmagatzemat en un determinat nombre d'aquests **blocs lògics** (i no necessitarà preocupar-se de com s'emmagatzemen físicament als diferents dispositius reals). Per exemple, un fitxer de **2000 bytes** ocuparà **2 blocs lògics** (1 Kbyte < 2000 bytes < 2 Kbytes), que podrien ser el **200** i el **201** (si estan els dos lliures).
El Controlador**,** per a dur a terme les peticions dels programes dels usuaris (o en sistemes reals, del Sistema Operatiu)**,** s'encarrega de comunicar-se, tant a **nivell elèctric** com de **protocol,** amb el dispositiu físic i informar-li a on es **situen físicament** els bytes dels **blocs lògics** sol·licitats. Suposem que el **controlador de disc** té connectat un **disc dur del tipus electromecànic** (suport magnètic). Aquest disc, físicament, està dividit en un conjunt de **C cilindres (numerats 0 al C-1)** de **P plats** cadascun (**numerats de la 0 a la P-1**) i amb **S sectors** per plats **(numerats 0 al S-1)**. Cada **sector** te una mida de **512 bytes** i es a on s'emmagatzemen (magnèticament) les dades (mida del disc: **(C*P*S)*512bytes**) . Per tant, el controlador al conèixer com s'emmagatzemen físicament les dades al disc, li pot informar (a través de missatges de control) en quins **cilindres, plats i sectors** estan situades les dades dels **blocs lògics** sol·licitats. Si, a més a més, el sistema implementa una interfície tipus SATA (Seria ATA), llavors el connector entre el controlador i el disc dur es un bus sèrie síncron d'alta velocitat amb 4 línies de dades (2 anada i 2 de tornada), compartint els missatges de dades i control aquestes línies de dades.
Per exemple, si el disc dur té una capacitat de **1 Mbyte**, el programador el veu con un **dispositiu lògic** de **1024 blocs** d'**un Kbyte** cadascun (1024 blocs * 1 kbyte = 1 Mbyte). Però físicament, aquest disc dur està format per **8 cilindres** amb **8 plats** per cilindre i amb **32 sectors** per plat (**(8 cilindres * 8 plats * 32 sectors)*512 bytes = 1 Mbyte**). Per tant, el **controlador del disc** al

conèixer aquesta informació, pot saber **quins sectors del disc** dur ocupen cada **bloc lògic**, en aquest cas:

Blocs lògics	Cilindre	plats	Sectors
0 al 15	**0**	**0**	**0,1** → bloc 0 ;…; **30,31** → bloc 15
16 al 31	**0**	**1**	**0,1** → bloc 16 ;…; **30,31** → bloc 31
		…..	
128 al 143	**1**	**0**	**0,1** → bloc 128 ;…; **30,31** → bloc 143
		…...	
704 al 719	**5**	**4**	**0,1** → bloc 704 ;…; **30,31** → bloc 719
		…..	
1008 al 1023	**7**	**7**	**0,1** → bloc 1008 ;…; **30,31** → bloc 1023

Descripció dels principals bits dels Registres d'Entrada/Sortida del controlador del disc dur:

El **bit 2** del **registre d'estat** del controlador de disc dur es posa a **1** quan **hi ha disponible un byte al registre de dades** (lectura de dades del disc dur) o quan aquest registre **està preparat per rebre un byte** (escriptura de dades al disc dur). Per **esborrar aquest bit**, **hi ha que escriure un 1 en el bit 2 del registre d'estat**. Per **activar localment la interrupció** del disc dur (per que el controlador la generi) hi ha que **escriure un 1 al bit 7 del registre de control**. Els **bits 6 i 5** del **registre de control** indiquen les següents accions: 00 → No hi ha **cap operació per realitzar-se**, **01** → Petició per fer una **escriptura al disc**, **10** → petició per fer una **lectura del disc.**

PROBLEMA 1.

Els principals passos que es fan amb una escriptura de dades (controlador I2C -> pantalla) son:

1) Configurar el controlador I2C (com a Mestre): Definir-lo com a dispositiu Mestre, (bit **MST** del reg. de control**), s**i envia o rep dades **(** bit **TX** del reg. de control), si s'activen les senyals **ACK** (bit **TXAC** del reg. de control**)** i l'activació de la generació d'interrupcions (bit **IICIE** del reg. de control**)**
2) Activar el controlador I2C: bit **IICEN** del reg. de control. **Comença la transmissió del missatge** amb la generació de la senyal **START** al bus **SDA** pel Mestre
3) Escriure en el registre de dades (bits 7 al 1) l'adreça de l'esclau (pantalla) i al bit 0 d'aquest registre, si es una operació de lectura (o escriptura).
4) Esperar l'activació del bit (**TCF)** del registre d'estat per assegurar-se que s'ha transmet la dada
5) Testejar el bit (**RXAC)** per assegura-se que s'ha rebut la senyal de **ACK** per part del esclau
6) Enviar les dades, escrivint-les en el registre de dades. Per cada dada enviada hi ha que repetir els **passos 4** (si es fa enquesta programada) i **5**.
7) Genera la condició de parada **(STOP)** desactivant el bit **(MST)** del registre de control

Escriure un fragment en llenguatge assemblador que primer inicialitzi el **controlador I2C** i que després envii **200 caràcters** (ASCII de 8 bits) a la pantalla alfanumèrica. Aquests caràcters estan

emmagatzemats en un **array de caràcters** situat a partir de l'adreça **0x9000** de la **memòria principal**.

a) Utilitzant la tècnica d'E/S programada amb espera de resposta.

```
;;1 Configuración controlador I2C, registro de control.
    1. LI R1, 0x00000200  ;Dirección inicio de los registros E/S del  controlador I2C
;; Primero se configura el funcionamiento del controlador (activando los bits
;; pertinentes).
    2. LI R6,0x00000030  ; Contenido reg. control, bits 7 al 0: 00110000.
                         ; No interrupciones (IICIE=0), Maestro (MST=1), Envío de datos (TX=1),
                         ; No señales ACK (TXAC=0, los ACK los situará el esclavo)
    3. SB R6, 2(R1)      ; Escribe el byte menos significativo de R6 al registro de control
                         ; 0x00000200 + 2  o 0x00000202 que corresponde a la dirección del reg.
                         ; de control del controlador I2C). ;R6(bits 7,..,0) -> [dir E/S (0x202)]
                         ; dir E/S significa direcciones de los registros de E/S (espacio E/S)
;; Después se activa el controlador.
    4. ORI R6, R6, 0x00000080 ;bit 7 de R6 a 1 (...XXXXXXXX | ..10000000 = ..1XXXXXXX)
                              ;Contiene el valor del bit IICEN=1
    5. SB R6, 2(R1)      ; Escribe el byte menos significativo de R6 al registro de control

;; 2) Primer byte mensaje: Situar la dirección I2C de la pantalla alfanumérica (esclavo)
;; (bits 7 a 1) y que es un envío de datos (bit 0 a 1) Maestro -> Esclavo (pantalla)
    6. LI R3, 0x0000002D ; dirección  I2C de la pantalla 0x2C ->  bits 7 al 1: 0010110 ; bit 0 a
                                                    ;1 (Envío) ; 00101101 (0x2D)
    7. SB R3, 0(R1) ; Escribe el byte menos significativo de R3 al registro de datos. Borra el
                    ; flag TCF (bit 7 reg. estado).
;; Guardar la dirección del primer elemento del array de caracteres de la memoria en R2
    8. LI R2, 0x0009000 ; dir. primer elemento del array de caracteres
    9. ADDI R5, R2, 200 ; dir. último elemento del array de caracteres (guardado en R5) .
;; Envío 200 caracteres a la pantalla alfanumérica
;; 3) Espera registro de datos este libre para el siguiente envío, bit  7 del registro de
;; estado a 1 (flag TCF)
espera:
    10. LB R3, 1(R1)  ; Lectura del reg. de estado (dir. 0x00000201), byte menos significativo
                      ; de R3 (bits (7,..,0) contiene el valor de  este registro
    11. ANDI R4,R3,0x00000080 ; bit 7 de R4 contiene el valor del flag TCF
                              ; ( ..1000.. & ..XXXX.. = ..X000...)
    12. BEQU R4, X0, espera  ; Si R4 = 0 , flag TCF = 0 -> Reg. de datos no preparado para
                             ; envío,  volver a leer reg de estado.
;; 4) Comprobar envío ACK del Esclavo
    13. ANDI R3,R3, 0x00000001 ; bit 0 de R3 contiene el valor del flag RXAC
    14. BEQU R3, X0, error_ACK  ; Si R3 = 0 , flag RXAC = 0-> Error ACK salir
;; 5) Comprobar si se han leído todos los caracteres.
    15. BGEU R2, R5, fin_transmision
;; 6)  Leer el dato de la posición del vector de caracteres y escribirlo al reg. de datos.
```

```
    16. LB R3, 0(R2) ; Leer el el dato de la posición del array de caracteres
                     ;R3(bits 7,..,0) <- [ mem (R2)]
    17. SB R3, 0(R1) ;  Escribir el byte menos significativo de R3 al registro de datos. Borra el
                     ; flag TCF
    18. ADDI R2,R2,1 ;Dirección siguiente elemento del array de caracteres (son bytes).
;; 5) Leer  siguiente carácter
    19. J espera
;; 7) Generar la señal de STOP (MST =0 y TX =0) para indicar final de transmisión
fin_transmision:
    20. ANDI R6,R6, 0xFFFFFFCF ; Solo se ponen a 0 los bits 5 y 4 del registro de control (el
                               ; contenido del registro de control esta en R6, ver principio)
    21. SB R6, 2(R1)
        ;;  --- Resto del código ---

error_ACK:
        ;;Acciones respecto al tratamiento del error.
```

b) Utilitzant la tècnica de E/S per interrupcions

```
;Variables globales:
; dir es del tipo entero (4 bytes) y contiene la dirección del elemento actual del array de datos

;; **********  Programa principal (main de C)
main:
    1.  LI R1, 0x9000 ; R1 contiene la dirección inicial del array de datos
    2.  LLA R2, dir
    3.  SW R1, 0(R2)     ;variable dir contiene la dirección del primer elemento del array

;;Configuración controlador I2C, registro de control.
;;Para el envío del primer byte del mensaje, las dirección del esclavo, no activamos las int
    4.  LI R1, 0x00000200  ;Dirección inicio registros controlador I2C
    5.  LI R6,0x00000030  ; Contenido reg. control, bits 7 al 0: 00110000.
                ; de momento NO interrupciones, Maestro, Envío de datos, No señales ACK
    6.  SB R6, 2(R1)     ; Escribe el byte menos significativo de R6 al registro de control
    7.  ORI R6, R6, 0x000000080 ;  bit 7 de R6 a 1
    8.  SB R6, 2(R1)     ; Escribe el byte menos significativo de R6 al registro de control
                         ; para activar el controlador. Empieza transmisión
;; Primer byte mensaje: Situar la dirección I2C de la pantalla alfanumérica
;; (bits 7 a 1) y que es un envío de datos (bit 0 a 1) Maestro -> Esclavo (pantalla)
    9.  LI R3, 0x0000002D ; dirección  I2C de la pantalla 0x2C ->  bits 7 al 1: 0010110 ; bit 0 a
                          ;1 (Envío)
    10. SB R3, 0(R1) ; Escribe el byte menos significativo de R3 al registro de datos. Borra el
                     ; flag TCF (bit 7 reg. estado).
    11. ORI R6, R6, 0x00000040 ;  bit 6 de R6 a 1
    12. SB R6, 2(R1)     ; Escribe el byte menos significativo de R6 al registro de control
                         ; Activar la generación de interrupciones para asegurarse que no se
                         ; activa ninguna durante el envío del primer mensaje.
```

```
infinito:
    13. WAI  ; Wait An Interrupt.  Duerme la CPU (ahorro de energía) a la espera de
             ; una  interrupción  que la despierte.

;; O instrucciones de otras tareas.

    14.  B infinito

 ;; ********** RSI asociada a la interrupción del I2C
rsi_I2C:
    15. LI R1, 0x00000200  ;Dirección inicio registros controlador I2C
    16. LB R2, 1(R1)  ; Lectura del reg. de estado (dir. 0x00000201), byte menos significativo
                      ; de R2 contiene el valor de  este registro
    17. ANDI R2,R2,0x00000001 ; bit 0 de R2 contiene el valor del flag RXAC
    18. BEQU R2, X0, salir_RSI_I2C  ; Si R2 = 0 , flag RXAC = 0-> Error ACK salir
;; Leer el dato de la posición del vector de caracteres y escribirlo al reg. de datos.
    19. LLA R2, dir
    20. LW R3, 0(R2)  ; R3 tiene el contenido de la variable dir. Dirección del elemento  actual
                      ; del array de  caracteres.
    21. LI  R4, 0x90C8 ; R4 tiene la dirección del último elemento del array: 0x9000 + 0xC8
                       ;(200 en hexadecimal) = 0x90C8
    22. BGEU R3,R4,  salir_RSI_todos  ; Si se han enviado todo los elementos del array, salir
                                      ; dir. elemento actual array >= dir. último elemento array
    23. LB R4, 0(R3) ; Leer el el dato de la posición del array de caracteres.
    24. SB R4, 0(R1) ;  Escribir el byte menos significativo de R3 al registro de datos. Borra el
                     ; flag TCF
    25. ADDI R3,R3,1 ;Dirección siguiente elemento del array de caracteres (son bytes).
    26. SW R3, 0(R2)  ; Guardo esta direccionan en la variable dir
    27. J salir_RSI_I2C
 salir_RSI_todos :
 ;;Generar la señal de STOP (MST =0 y TX =0) para indicar final de transmisión
    28. LB R2, 2(R1)
    29. ANDI R2,R2, 0xFFFFFFCF ; Solo se ponen a 0 los bits 5 y 4 del registro de control (el
                               ; contenido del registro de control esta en R6, ver principio)
    30. SB R2, 2(R1)
salir_RSI_I2C:
    31. MRET                   ;Retorno de Rutina de Interrupción
```

c) Si volguéssim que mentre s'executa una RSI no es pugues atendre cap altra interrupció (sistema **no apropiatiu** o **sistema d'interrupcions inhibit**). Quins canvis s'haurien de fer a la RSI anterior?

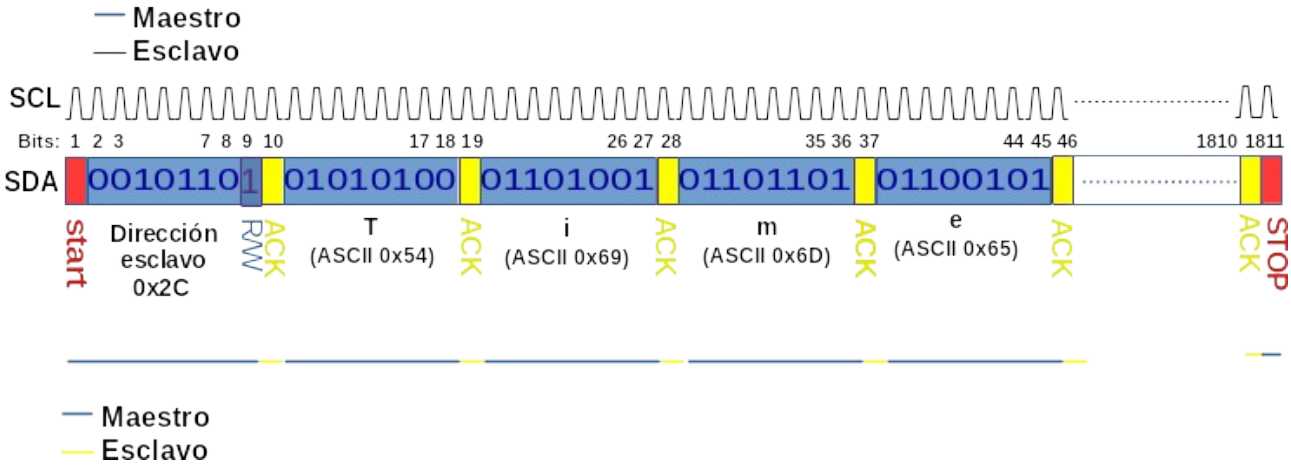
Para inhibir el sistema de interrupciones deberíamos poner el bit 3 del registro MSTATUS a 0, de esta forma se desactivarían globalmente las interrupciones y la CPU no aceptaría ninguna interrupción más. Por lo tanto, se debería añadir a la RSI las siguientes instrucciones:

```
 ;; ********** RSI asociada a la interrupción del I2C
rsi_I2C:
        ;; A la entrada de la RSI inhibimos las interrupciones globalmente
```

```
        ANDI  MSTATUS,  MSTATUS, 0xFFFFFFF7 ;bit 3 de MSTATUS a 0, resto de bits NO
                                             ;se  modifican
        ;;
        ;;   Código de la RSI
        ;;
        ;; A la salida  de la RSI volvemos a activar  las interrupciones globalmente
salir_RSI_I2C:
        ORI  MSTATUS,  MSTATUS, 0x00000008 ;  bit 3 de MSTATUS a 1 , resto de bits NO
                                           ; se  modifican
        MRET                    ;Retorno de Rutina de Interrupción
```

PROBLEMA 2

Si els 4 primers caràcters del array, situat en memòria, son **'T', 'i','m'** i **'e',** mostreu el missatge (fins als 4 primers caràcters i el final d'aquest missatge) que enviarà el controlador I2C, pel cable **SDA** a la pantalla alfanumèrica.



PROBLEMA 3.

Suposar que el temps des de que el controlador I2C activa la línia d'interrupció fins que es comença a la RSI és de 50 ns. Calcular el temps que dedica la CPU a transferir els 200 caràcters del array de caràcters a la pantalla alfanumèrica.

a) Utilitzant la tècnica de E/S programada amb espera de resposta vista al exercici 1 a)

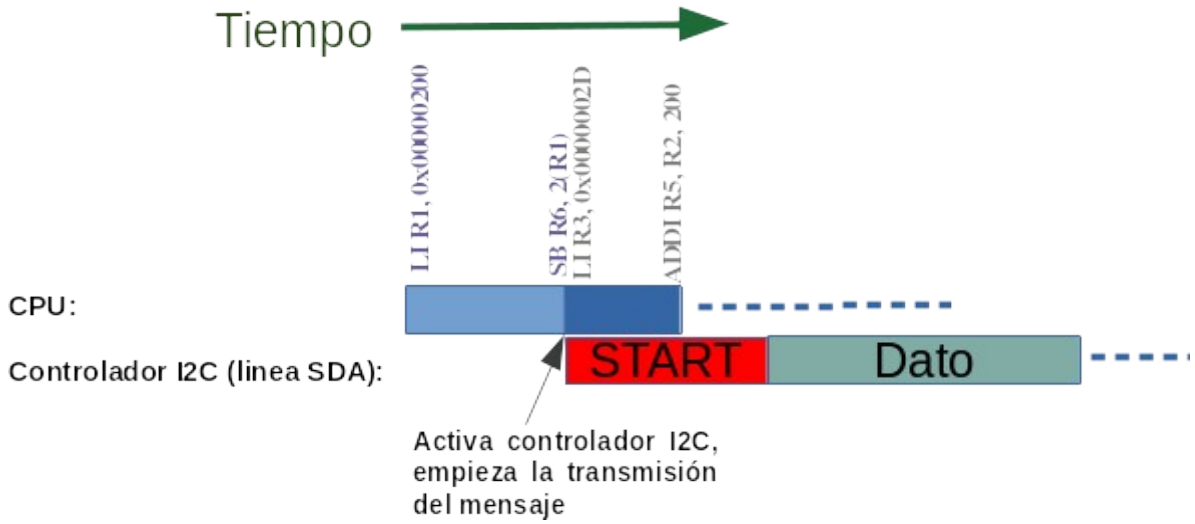
La CPU estará dedicada todo el tiempo a realizar la transmisión de los caracteres a la pantalla.

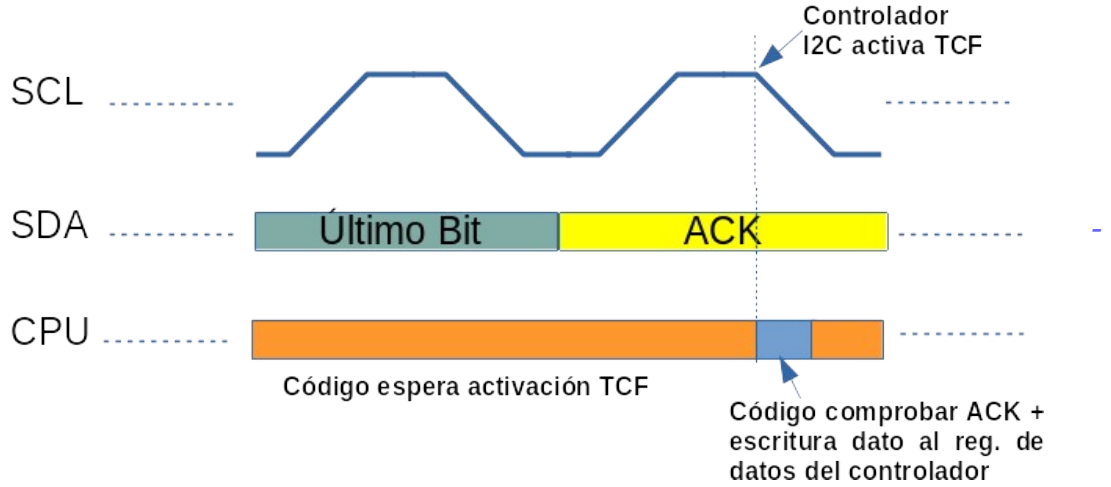
- **Tiempo ciclo CPU:**

$1/(200 * 10^6) = 5* 10^{-9}$ seg. o 5 ns

- ***Tiempo ciclo SCL:***

Si velocidad de transmisión del bus es de 3,4 Mbits/s el periodo de transmisión de 1 bit será: $1/ (3,4 * 2^{20})$ seg. = $2,8 *10^{-7}$ seg. o 280 ns.





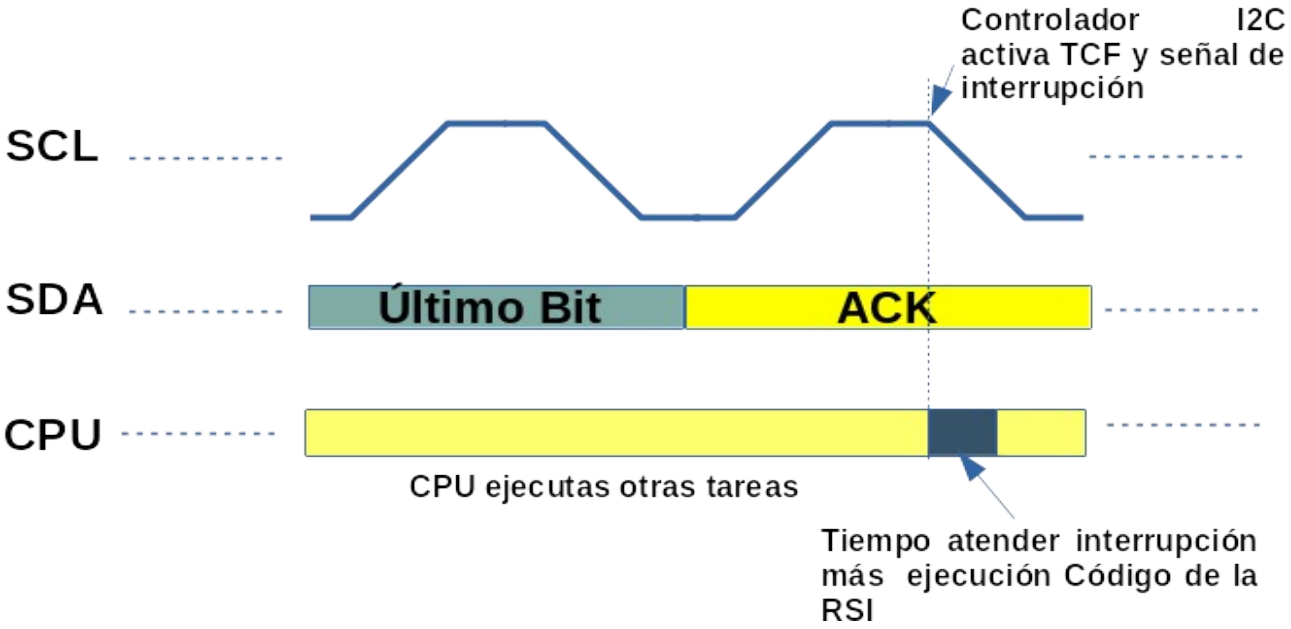
Tiempo ocupada CPU: Tiempo inicialización controlador + Tiempo transmisión mensaje (1 byte dirección del esclavo + 200 caracteres =201)

- ***Tiempo inicialización controlador (instrucciones 1 a la 5) :***
  (Numero de instrucciones inicialización controlador y del envío primer dato al reg de datos * Tiempo ciclo CPU) = 5 * 5 ns = 25 ns
- ***Tiempo transmisión mensaje (201 bytes) :***
  Bits del mensaje: **1 bit se START + 8 bits primer mensaje (7 dirección esclavo + 1 R/W) + 1 bit ACK del esclavo + 200* (8 bits datos + 1 ACK esclavo) + 1 bit STOP**
  Total bits: **10 + (200* 9) + 1 = 10 + 1800 +1 = *1811 bits.***
  Tiempo total : 280 ns * 1811 bits = 507000 ns o 507 us

Tiempo ocupada CPU: 25 ns + 507 us = 507,025 us

b) Utilitzant la tècnica de E/S per interrupcions vista al exercici 1 b)

La CPU se ocupa de transmitir un carácter (a través de la RSI) sólo cuando se produce una interrupción (reg. datos del controlador I2C listo para enviar dato). La RSI será llamada 201 veces (200 para cada uno de los 200 caracteres y 1 para el ultimo ACK que envía el STOP.).



El tiempo que estará ocupada la CPU para la transmisión :

Tiempo ocupada CPU: Tiempo inicialización controlador + Tiempo instrucciones primer mensaje (mensaje de envío dir. esclavo al reg. de datos) + Tiempo ejecución 200 RSIs de transmisión de datos + Tiempo ultima RSI (comprobación último ACK)

$$Tiempo\ para\ 1\ RSI = \left(\frac{Total\ instrucciones\ RSI}{MIPS\ de\ la\ CPU} + tiempo\ atender\ a\ la\ interrupción\right)$$

- ***Tiempo inicialización controlador + Tiempo instrucciones primer mensaje (mensaje de envío dir. esclavo al reg. de datos) : 12 instrucciones (de la 1 a la 12) * 5 ns = 60 ns***
- ***Tiempo ejecución 200 RSIs de transmisión de datos + Tiempo ultima RSI (comprobación último ACK:***
  - ***Tiempo ejecución 200 RSIs de transmisión de datos:***
    200 * (14 instrucciones (de la 15 a la 27 y la 31) * 5 ns + 50 ns) = 200 (70 ns + 50 ns) = 2,4 *10 ^{4} ns o 24 us
  - ***Tiempo ultima RSI (comprobación último ACK) :***
    (12 instrucciones (de la 15 a la 22 y de la 28 a la 31) * 5 us) + 50 ns = 60 ns + + 50 ns = 110 ns

Tiempo ocupada CPU: 0,060 us + 24 us + 0,110 us = 24,170us

c) Amb la tècnica de E/S per interrupcions. Quin percentatge del temps d'execució de la CPU te per dedicar a altres tasques mentre dura la transmissió dels 200 caràcters?

- Tiempo total transmisión con interrupciones respecto a la CPU= Tiempo inicialización controlador + Tiempo transmisión de 201 datos (507 us) :
- Tiempo inicialización controlador :
8 instrucciones (de la 1 a la 8) *5 ns= 40 ns
- Tiempo total transmisión con interrupciones respecto a la CPU:
40 ns + 507 us = 507,040 us

% tiempo ejecución CPU libre: *((507,040 ns - 24,170 ns) *100)/507,040 ns =95,24 %*

PROBLEMA 4.

Escriure un fragment en llenguatge assemblador que primer inicialitzi el **controlador del disc dur** i que després envii **1024 bytes, emmagatzemats** en un buffer situat a partir de l'adreça 0x3000 de la memòria (i que emmagatzema el contingut d'un fitxer), al **bloc 100 del disc dur**.

a) Utilitzant la tècnica d'E/S programada amb espera de resposta.

```
        LI R1, 0x3000                   ; R1, Dirección base del buffer de memoria= 0x3000
;;Inicialización controlador E/S disco.
        LI R2, 0x000000064    ; 0x64 → 100 en decimal (id del bloque)
        LI R3, 0x000000208    ; Dirección inicio registros controlador disco
        ADDI R5,R1, 1024      ; R5 tiene la dirección final del buffer  0x3000 + 0x400 (1024)
        SH R2, 3(R3)            ; Sitúa el identificador del bloque al registro Blk (16 bits)
                                ;del controlador.
                                ;R2 (bits 15,..,0) -> [dir E/S (0x20C) - dir E/S(0x0x20B)]
        LB R2, 2(R3)          ; Leer el registro de control a R2 (byte bajo)
        ANDI R2, R2, 0xFFFFFF3F    ; 0x3F → 00111111, AND bits 7 y 6 a 0,  resto se mantiene
        ORI R2, R2, 0x00000020      ; 0x20 →  00100000, bit 5 → 1, resto se mantiene
                            ; R2 tiene ..001XXXXX, bits 7,6 y 5→ 001, No interrupciones locales
                            ;(bit 7 =0) y se realiza una escritura (bits 5,6 =01).
                            ; X: Valor original (antes de OR y AND) puede ser 1 o 0.
        SB R2, 2(R3)            ; Escribir el contenido de R2 (byte bajo)  al reg. Control

;;Escritura memoria → disco.
        LI R2,0x00000004            ; R2  máscara del bit 2 (0x4→00000100) registro de estado
wait_byte:
        LB R4, 1(R3)              ; Sitúa el contenido del reg. de estado  en  R4 (byte bajo)
        AND R4, R4, R2              ; Solo se mira el bit 2, el resto de bits a 0
        BNEU R4, R2, wait_byte ; Espera que el bit 2 del reg. de estado se  ponga a 1
        SB R2, 1(R3)                  ; Borra el bit 2 del reg. de estado  (escribe un 1 en este bit)
        LB R2, 0(R1)                ; Leer el carácter de la posición actual del buffer
        SB R2, 0(R3)                  ; Escribir el carácter al registro de datos
        ADDI R1,R1,1                  ; Siguiente posición del buffer
        BLTU R1,R5, wait_byte  ; Si se han leído menos de 1024 caracteres,  volver a esperar
;;Informar final transacción memoria → disco
        LB R2, 2(R3)                ; Leer el registro de control.
        ANDI R2, R2, 0xFFFFFFF1F ; 0x1F→000 11111, AND bits 7,6 y 5 a 0,  resto se mantiene
                            ; R3 tiene  ...000XXXXX, bits 7,6 y 5→ 000, No interrupciones
                            ;locales (bit 7 =0) y no hay operación (bits 5,6 =00).
        SB R2, 2(R3)                  ; Escribir el contenido de R3 (byte bajo)  al reg. Control
```

b) Utilitzant la tècnica de E/S per interrupcions, escriure sols la RSI i la inicialització del controlador d'E/S del disc (no el codi que instal·la la RSI associada al disc).

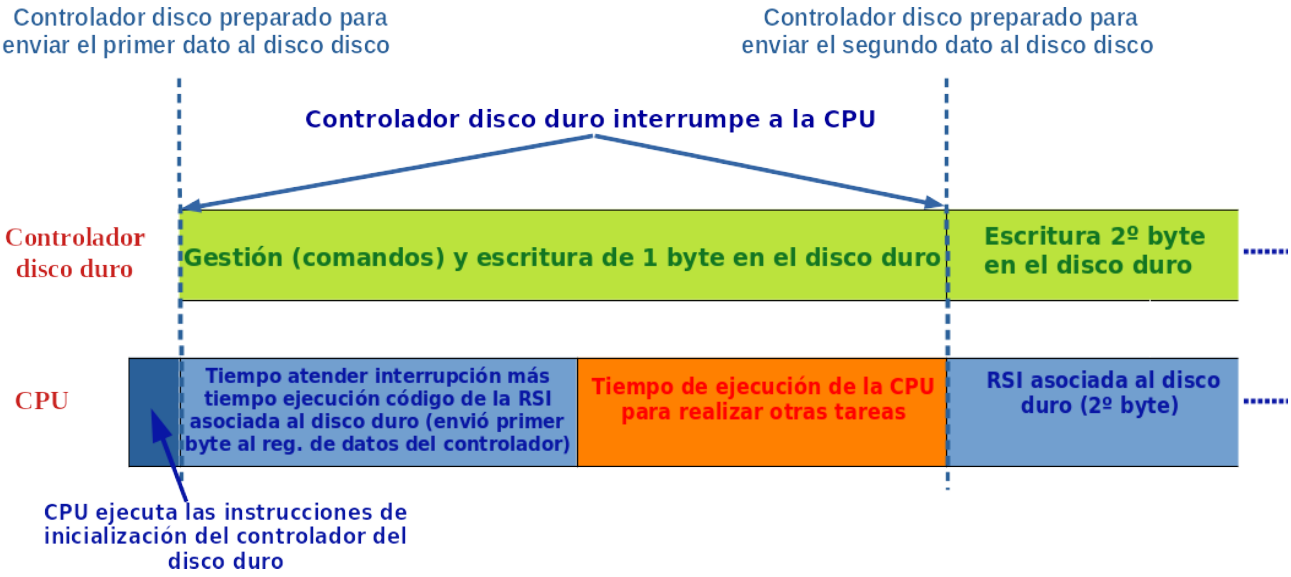
```
;Variables globales:
```

```
; dir es del tipo entero (4 bytes) y contiene la dirección de la posición del buffer de memoria

;; **********  Programa principal (main de C)
main:
        LI R1, 0x3000              ; R1, Dirección base del buffer de memoria= 0x3000
        LLA R2, dir                ; Var dir contiene dirección actual del  buffer de memoria
        SW R2, 0(R1)               ;Se inicializa con la dirección inicial del buffer
;;Inicialización controlador E/S disco.
        LI R1, 0x000000208    ; Dirección inicio registros controlador disco
        LI R2, 0x000000064    ; 0x64 → 100 en decimal (id del bloque)
        SH R2, 3(R1)               ; Sitúa el identificador del bloque al registro Blk (16 bits)
                                   ;del controlador.
        LB R2, 2(R1)           ; Leer el registro de control a R2 (byte bajo)
       ANDI R2, R2, 0xFFFFFFBF    ; 0xBF → 10111111, AND bit 6 a 0,  resto se mantiene
        ORI R2, R2, 0x0000000A0      ; 0xA0 →  10100000, bits 7 y 5 → 1, resto se mantiene
                              ; R2 tiene ...101XXXXX, bits 7,6 y 5→ 101, SI interrupciones locales
                              ;(bit 7 =1) y se realiza una escritura (bits 5,6 =01).
                              ; X: Valor original (antes de OR y AND) puede ser 1 o 0.
        SB R2, 2(R1)               ; Escribir el contenido de R3 (byte bajo)  al reg. Control

infinito:
         WAI        ; Wait An Interrupt.  Duerme la CPU (ahorro de energía) a la espera de
                    ; una  interrupción  que la despierte.

        ;; O instrucciones de otras tareas.

         B infinito
 ;; ********** RSI asociada a la interrupción del disco

rsi_disco:
        LI R3, 0x000000208      ; R3 tiene la dirección inicio registros controlador disco
        LLA R1, dir                   ;R1 tiene la dirección de la variable dir
        LW R2, 0(R1)                        ;R2 tiene la dirección posición actual buffer
        LI R4,0x3400      ; R4 tiene la dirección final del buffer  0x3000 + 0x400 (1024)
        BGE  R2,R4, todos_rsi
        LI R4,0x4                   ; R4  máscara del bit 2 (0x4→00000100) registro de estado
        SB R4, 1(R3)                  ; Borra el bit 2 del reg. de estado  (escribe un 1 en este bit)
        LB R4, 0(R2)                ; Leer el carácter de la posición actual del buffer
        SB R4, 0(R3)                  ; Escribir el carácter al registro de datos
        ADDI R2,R2,1                  ; Dirección de la siguiente posición del buffer
        SW R2, 0(R1)                  ; Guardar esta posición en la variable dir
        J fin_rsi
todos_rsi:
;;Informar final transacción memoria → disco
        LB R2, 2(R3)                ; Leer el registro de control.
      ANDI R2, R2, 0xFFFFFF1F  ; 0x1F → 00011111, AND bits 7,6 y 5 a 0,  resto se mantiene
                              ; R3 tiene ...000XXXXX, bits 7,6 y 5→ 000, No interrupciones
                              ;locales (bit 7 =0) y no hay operación (bits 5,6 =00).
        SB R2, 2(R3)                ; Escribir el contenido de R3 (byte bajo)  al reg. Control
```

```
fin_rsi:
        MRET                    ;Retorno de Rutina de Interrupción
```

PROBLEMA 5.

a) Suposar que el temps des de que el **controlador de disc dur** activa la línia d'interrupció fins que es comença a la RSI és de 50 ns. i que la velocitat de transmissió (anomenada mitjana) del disc dur es de 5 MBytes/s (la de 200 MBytes/s es la màxima, aconseguida en modes especials com el DMA). Calcular el temps que dedica la CPU a realitzar la transferència de 1024 bytes del exercici anterior per la tècnica de E/S d'interrupcions.



-Tiempo que dedica la CPU a la ejecución de 1 RSI:

T. ejecución instrucciones RSI + Tiempo atender int.: (13 * 5 ns + 50 ns) = 65 ns + 50 ns = **115 ns.**

La última RSI ejecutada seria la que la variable **dir** almacena la dirección **0x3400** con 9 instrucciones ejecutadas, pero no la consideramos (si lo hiciéramos, añadiría 9 * 5 ns = 45 ns a la escritura total)

- Tiempo de ejecución que dedica la CPU a la transacción:

T. inicialización controlador E/S)+ 1024 * (t. ejecución RSI)= (10 instrucciones * 5 ns) + 1024*115 ns= 50 ns + 117800 ns. = **117850 ns.**

b) Podríem realitzar aquesta transferència utilitzant E/S per interrupcions? (pista, penseu en quan triga el controlador del disc dur en estar preparat per rebre un byte i el temps total d'execució de la RSI)

Solución 1:

-Disco duro, 5Mbytes/seg. Escribe (o lee) un byte del disco cada: $1/(5*2^{20})$= 190 ns.

Si que se podría: Cada RSI tarda 115 ns en leer un byte y el disco duro transmite 1 byte cada 190 ns (la CPU tiene 190 ns-115 ns=75 ns de tiempo de ejecución libre entre interrupciones, que representan 75 ns / 5 ns = 15 instrucciones).

Solución 2:

- Tiempo total transferencia respecto CPU (inicialización controlador (50ns) + tiempo transferencia del disco):

- Tiempo transferencia del disco: 2^{10} / (5* 2^{20}) = 1,95 * 10^{-4} seg. o 195 us (195000 ns)

- Tiempo total transferencia respecto CPU= 50 ns + 195000 ns = **195050 ns**

***La transacción ocupa más tiempo* (195050 ns) que el *tiempo de ejecución* que dedica *la CPU* a la misma *por interrupciones* (117850 ns). Por lo tanto la CPU dispone de tiempo de ejecución libre para dedicarlo a otras tareas.**

(Debido a que el tiempo de inicialización del controlador es muy pequeño respecto al de transmisión: 50 ns respecto a 195 us, se puede despreciar este tiempo de inicialización del controlador)

c) Quina seria la funció lògica que generaria la senyal CS del controlador del disc dur?

	A31..A12	A11..A8	A7..A4	A3..A0
0x00000208	0000	0010	0000	1000
0x00000209	0000	0010	0000	1001
0x0000020A	0000	0010	0000	1010
0x0000020B	0000	0010	0000	1011
0x0000020C	0000	0010	0000	1100

CS=A15* A14* A13* A12* A11* A10* A9 A8* A7* A6* A5* A4* A3 MEM/IO*

* Significa negado.

PROBLEMA 6.

a) Escriure un fragment de codi en llenguatge assemblador per a programar el controlador de DMA i que transfereixi **8192 bytes** situats a partir del **bloc 1000** del disc (ocupen 8 blocs, del 1000 al 1007) a un **buffer de memòria que comença a l'adreça 0xA000 .** Suposar que la transferència es posa en marxa al escriure el valor **0Fh** en el registre de control del DMA.

```
1.  LI R1, 0x000000300   ; Dirección inicio registros controlador DMA
2.  LI R2, 0x0000A000    ; Dirección de memoria donde empiezan los datos
3.  SW R2, 0(R1)         ; Situar esta dirección en el registro de dirección del DMA (32 bits)
                         ;R2 (bits 31,..,0) -> [dir E/S (0x303)- dir E/S (0x300) ]
4.  LI R2, 0x00002000    ;  Numero de bytes totales a transmitir (0x2000 → 8192 dec.)
5.  SH R2, 4(R1)         ; Situar esta dirección en el registro contador del DMA (16 bits)
6.  LI R2, 0x000003E8    ;  Numero del bloque del disco duro  ( 0x3E8 → 1000 dec.)
7.  SH R2, 6(R1)         ; Situar esta dirección en el registro de bloque del DMA (16 bits)
8.  LI R2, 0x000000F0    ;  Valor inicio transferencia (8 bits)
9.  SB R2, 8(R1)         ;Situar este valor en el registro de control. Inicia la transferencia
```

b) Si la CPU tingues l' **espai d'adreces de E/S separat del d'adreces de memòria principal.** Que canviaria respecte al codi anterior?

Si la CPU tuviera espacio de direcciones de E/S separado del de direcciones de memoria principal, implicaría que tendría unas instrucciones especiales para acceder a los registros de E/S (diferentes de las que acceden a memoria principal). Estas instrucciones son:

IN[B/H/W] Rd, desplazamiento(Rm): **Leer un dato (B:byte,H:halfword, W:word) de un registro de E/S (cuya dirección origen es Rm + desplazamiento) y situarlo en los bytes correspondientes del registro Rd (ver explicación L[B/H/W]).**
OUT[B/H/W] Rd, desplazamiento(Rm): **Escribir un dato (B:byte,H:halfword, W:word), contenido en los bytes correspondientes del registro Rd, a un registro de E/S (cuya dirección origen es Rm + desplazamiento).**

Entonces se tendrían que cambiar todas las instrucciones de L[B/H/W] (lectura) o S[B/H/W] (escritura) que acceden a las direcciones de los registros de E/S por IN[B/H/W] (lectura) o OUT[B/H/W] (escritura).

linia	Cambio de instrucciones
3	SW R2, 0(R1) por OUTW R2, 0(R1); Situar esta dirección en el registro de dirección ;del DMA (32 bits).
5	SH R2, 4(R1) por OUTH R2, 4(R1); Situar esta dirección en el registro contador del ;DMA (16 bits)
7	SH R2, 6(R1) por OUTH R2, 6(R1) ; Situar esta dirección en el registro de bloque del ;DMA (16 bits)
9	SB R2, 8(R1 por OUTB R2, 8(R1) ; Inicia la transferencia (8 bits)

Si en R1 está la dirección inicio de los registros controlador DMA, entonces una lectura del registro de control seria:
INB R2,8(R1)

PROBLEMA 7.

a) Cada vegada que el controlador DMA demana el bus a la CPU, aquesta li dona prioritat immediatament (mecanisme de cicle robat). Per a la transferència d'una dada (1 byte a la velocitat de transferència màxima del disc de **200 MBytes/s**) entre el controlador del disc i la memòria, s'utilitza el **bus de memòria durant 3 ns**, i es torna a cedir l'ús del bus a la CPU. Aquests 3 ns inclouen el temps de cessió del bus, el temps de la transferència pel bus, i el temps de recuperació del bus.
Calcular el percentatge del temps que dedica la CPU a l'operació de transferència per DMA dels **8 Kbytes** del exercici anterior. Suposar que la **CPU dedica 500 ns per a programar el DMA** (codi d'inicialització del DMA més coordinació amb el controlador del disc) i **500 ns per a reconèixer la finalització de la tasca del DMA** i que la **CPU no pot fer cap tasca** durant tot el temps en que el bus està ocupat per part del controlador DMA.

-Tiempo ocupación del bus por el DMA:
8192 (accesos de 1 byte) * 3 ns (tiempo ocupación bus para transmitir 1 byte) = 24576 ns o 24,5 us
- Tiempo ejecución CPU dedicado a la transferencia (programación y finalización DMA +

tiempo ocupación del bus por el DMA):

1 us (500 ns + 500 ns) + 24,5 us =25,5 us

- Tiempo total transferencia respecto CPU (programación y finalización DMA + tiempo transferencia del disco):

1 us + ((8*2^{10})/(200*2^{20})) s = 1 us + 39 us = 40 us

- Porcentaje de tiempo ocupado: 100 × (25,5 us / 40 us) = 64%

b) ¿Cuanto tiempo dispone la CPU para la ejecución de otras tareas después de la ejecución de una RSI del apartado b), la cual escribe un dato de 4 bytes del registro de estado a la posición del array?

- Tiempo ocupación del bus por el DMA:

1024 (accesos de 1 byte) * **3 ns** (tiempo ocupación bus para transmitir 1 byte) = **3072 ns o 3 us**

- Tiempo ejecución CPU dedicado a la transferencia (programación y finalización DMA + tiempo ocupación del bus por el DMA):

1 us (500 ns + 500 ns) + 3 us =4 us

- Tiempo total transferencia respecto CPU (programación y finalización DMA + tiempo transferencia del disco):

1 us +((1*2^{10})/(5*2^{20})) s = 1 us + 195 us = 196 us

- Porcentaje de tiempo ocupado: 100 × (4 us / 196 us) = 2%

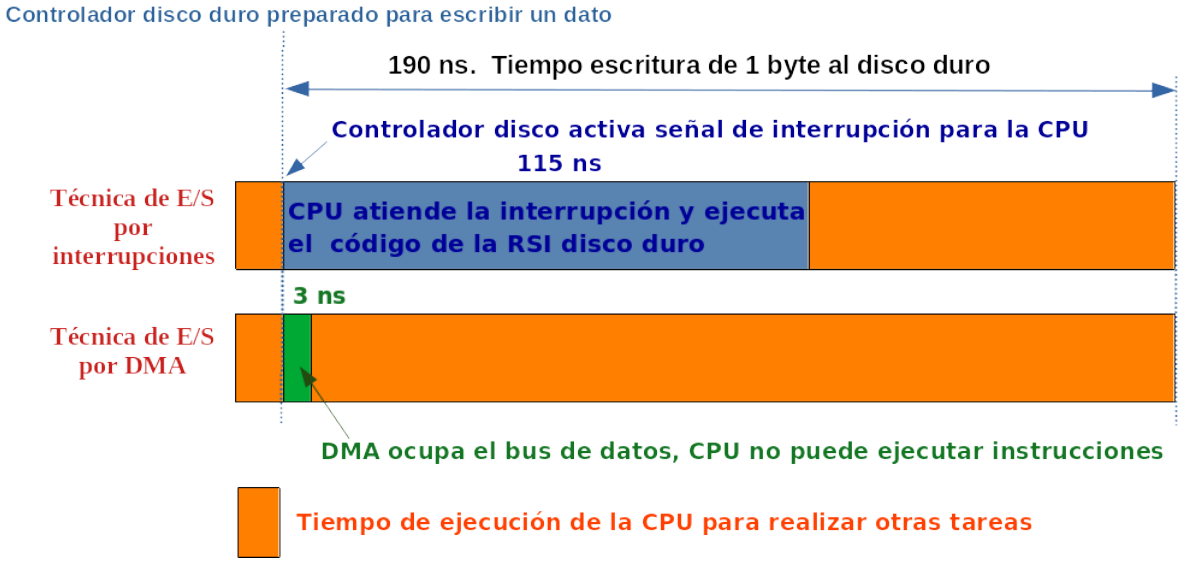
c) Comparar **% del temps d'execució dedicat per la CPU** a la transferència dels **1024 bytes del exercici 4**, al utilitzar el mètode de transferència per **DMA** o el d'**interrupcions.**

- Tiempo transferencia de los 1024 bytes por el disco: 195 us

- Por el método de transferencia con interrupciones, el % del tiempo de ejecución de la CPU es de (100 x 117,850 us) /195 us = 60,4%

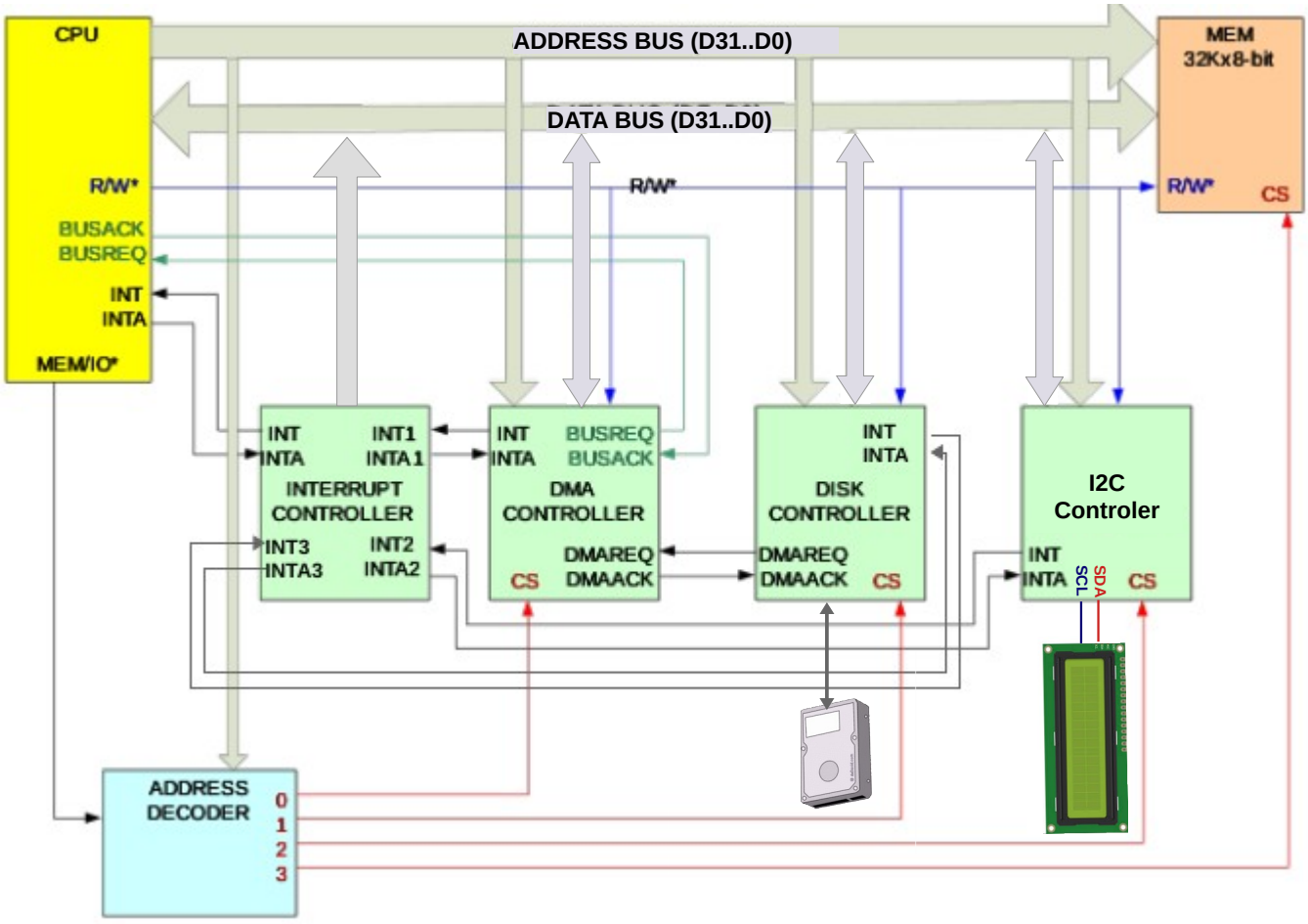
(T. total de transferencia respecto a la CPU seria = t. inicialización controlador + t. transferencia del disco = 195 us + 0,05 us, pero este último tiempo lo podemos despreciar)

- Por el método DMA el % del tiempo de ejecución de la CPU es solo de un 2%. Mucho mejor este último.



PROBLEMA 8.

a) Completar el diagrama de blocs del sistema computador utilitzant els elements descrits en la figura, afegint els senyals necessaris i les seves interconnexions, i indicant el seu sentit (entrada, sortida, o entrada/sortida). **En aquest apartat NO considerar la generació dels senyals CS.**



b) Generar la funció lògica de la línia CS del controlador DMA (sortida 0 Address decoder) suposant que:

- Els ports es troben a l'**espai compartit d'adreces de memòria (**descodificació completa).

Controlador DMA utiliza las direcciones desde la 0300h hasta la 0308h.

	A31..A12	A11..A8	A7..A4	A3..A0
0300h	0000	0011	0000	0000
0301h	0000	0011	0000	0001
0302h	0000	0011	0000	0010
0303h	0000	0011	0000	0011
0304h	0000	0011	0000	0100
0305h	0000	0011	0000	0101
0306h	0000	0011	0000	0110
0307h	0000	0011	0000	0111
0308h	0000	0011	0000	1000

CS=A15* A14* A13* A12* A11* A10* A9 A8 A7* A6* A5* A4*

- Els ports es troben a l'**espai separat d'adreces de E/S (**descodificació completa).

CS=A15* A14* A13* A12* A11* A10* A9 A8 A7* A6* A5* A4* MEM/IO*

* Significa negado.

PROBLEMA 9.

a) Escriure un fragment en llenguatge assemblador que realitzi les següents opcions:
1) Inicialitzi l'adreça inicial de la taula de vectors d'interrupció
2) Activi les interrupcions globalment i les de tots els controladors del sistema (activació local de les interrupcions pel controlador I2C, el del disc i el del DMA)

```
;; Inicialización de la dirección inicial de la tabla de vectores de interrupción
LI MTVEC , 0x00008000 ; Sitúa el valor de 0x8000 (dirección inicial tabla vectores de
                      ; interrupción) en MTVEC.
;; Activación global de las interrupciones
ORI MSTATUS, MSTATUS, 0x00000008 ; Sitúa el bit 3 de MSTATUS a 1, el resto no se
                                 ; modifican
;;Activación local de las interrupciones
ORI MIE, MIE,  0x0000A800 ; Sitúa a 1 los bits 15,13,11 de MIE , el resto no se modifican
```

b) Completar la taula (omplint les caselles ombrejades) que indica les passes a realitzar per a la gestió d'una interrupció del controlador I2C. Indicar quan estan actives i desactives els senyals INT e INTA.

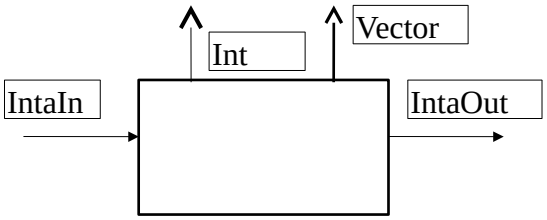
MÒDUL	BUS Adreces	BUS DADES	INT	INTA	INT2	INTA2	Comentari
I2C	------	------	----	----	**ACT**	DES	Es pot enviar un nou caràcter pel bus SDA i el controlador I2C activa el senyal INT
Cont. INT.			**ACT**	DES	ACT	DES	**El controlador d'interrupcions detecta la senyal INT2 i activa la senyal INT (per la CPU)**
CPU	------	------	ACT	DES	ACT	DES	**Acaba instrucció en curs i detecta INT**
CPU	-------	-------	ACT	**ACT**	ACT	DES	CPU activa INTA i dóna inici al cicle de reconeixement d'interrupció
Cont. INT.			ACT	ACT	ACT	**ACT**	**Detecta l'activació de la senal INTA i activa INTA2 pel I2C**
I2C	-------		ACT	ACT	**DES**	ACT	**Reconeix INTA2 i desactiva el senyal INT2**
Cont. INT.		**VECTOR (valor 11)**	**DES**	ACT	DES	ACT	**Situa l'identificador del vector d'interrupcions del I2C i desactiva el senyal INT**

CPU	-------	**VECTOR**	DES	**DES**	DES	ACT	CPU llegeix l'identificador del vector d'interrupció del bus de dades, el situa en el reg. **MCAUSE** i desactiva INTA
Cont. INT.			DES	DES	DES	**DES**	**Desactiva el senal INTA2**
CPU	**Dir. Taula Vectors**[1]	------	DES	DES	DES	DES	**Calcula adreça de la taula de vectors d'interrupcions que conté l'adreça de la RSI associada al I2C i la posa al bus**
Memòria	-------	**DIR. RSI**[2]	DES	DES	DES	DES	Memòria contesta amb l'adreça de la RSI associada al I2C
CPU	------	------	DES	DES	DES	DES	**Salva registres, PSW i adreça de retorn**
CPU	-------	------	DES	DES	DES	DES	Modifica el Comptador de Programa per a què contengui l'adreça de la primera instrucció de la RSI

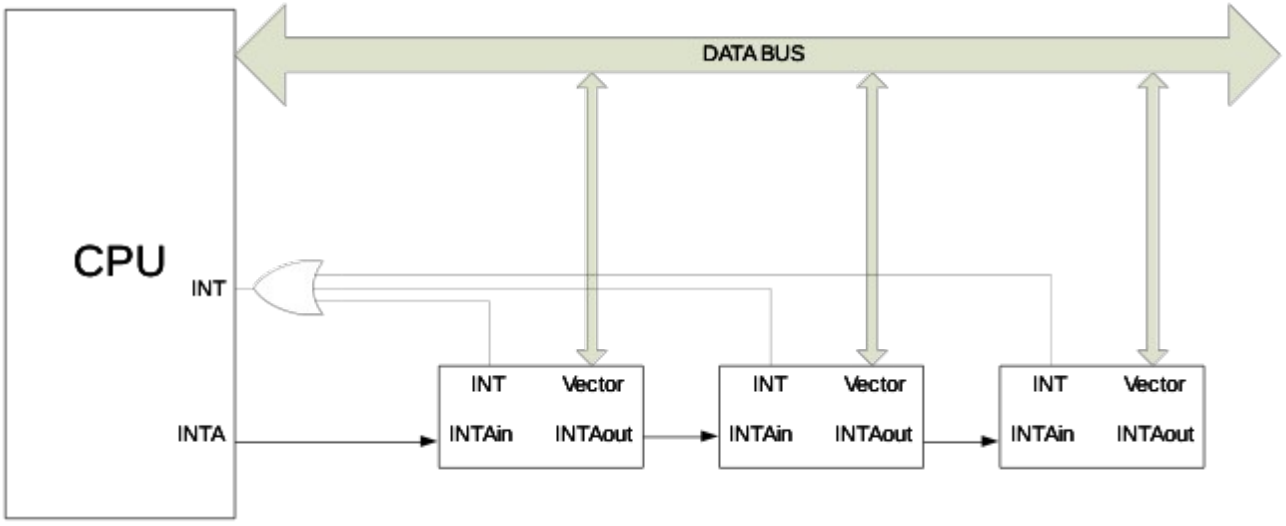
[1] Las direcciones de memoria **32812 a la 32815 (0x802C-0x802F)** de la tabla de vectores de interrupción, contienen la dirección de la RSI (4 bytes) asociada al controlador I2C.
Por lo tanto, la dirección que sitúa al bus de direcciones la CPU es **32812 (0x802C) -> 32768** (ó **0x8000,** dirección inicio tabla vectores de interrupción) **+ (11** (identificador de la tabla de vectores de interrupciones asociado al I2C) *** 4** (direcciones de 32 bits) **).**
[2] Supongamos que el **código de la RSI asociada al controlador I2C** está situado a partir de la dirección de memoria **0x8512. Entonces las direcciones de memoria 32812 a la 32815 (0x802C-0x802F) de la tabla de vectores de interrupción** contendrán **0x8512 (la dirección de la RSI asociada al controlador I2C).**

PROBLEMA 10.

Tenim tres mòduls d'entrada/sortida amb capacitat d'interrupció com el següent:

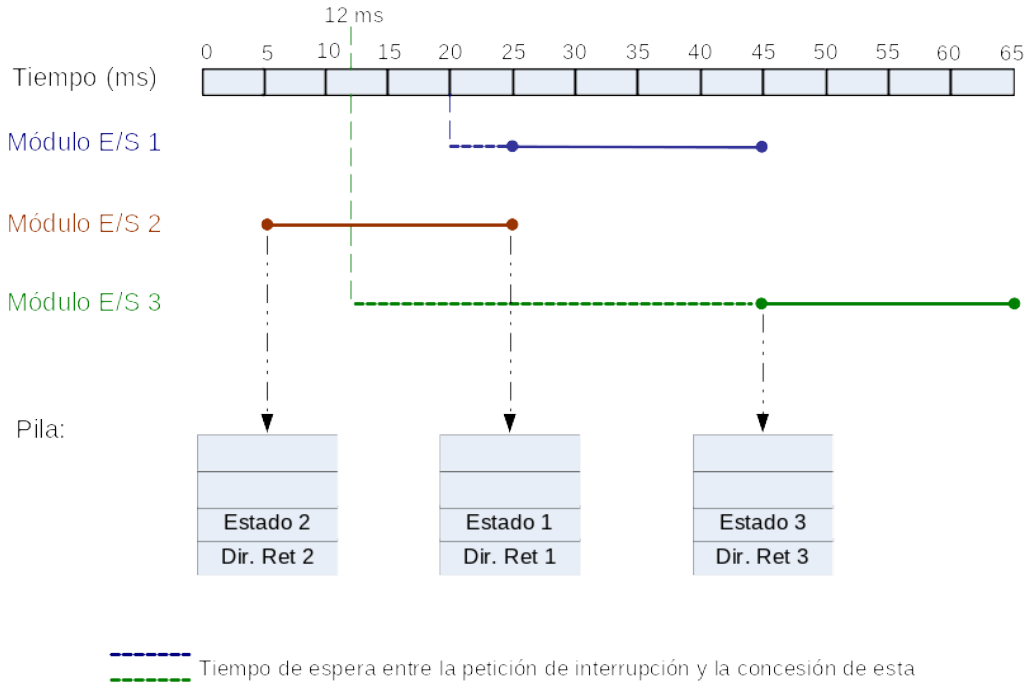


a) Mostrar la connexió dels mòduls amb la CPU, considerant que la CPU té una única línia de petició d'interrupció i una única línia de reconeixement.

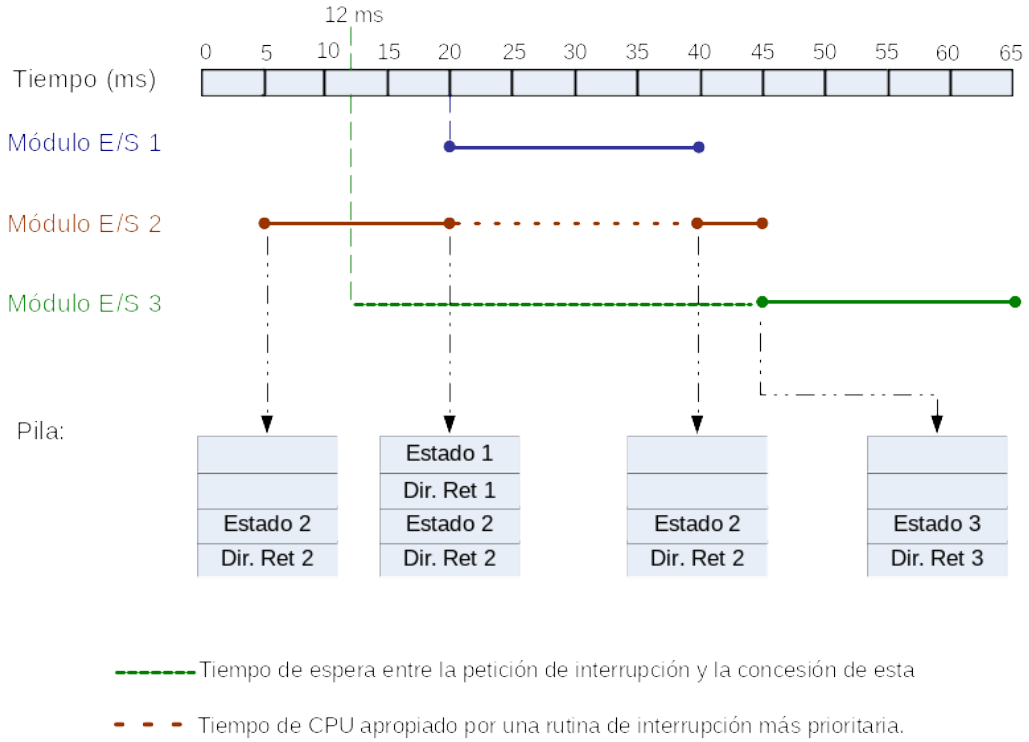


b) Si cada una de les rutines d'atenció als mòduls d'E/S triguen 20 ms (considerar que el temps de reconeixent de la interrupció està inclòs), mostrar l'evolució de la pila si la seqüència d'arribada de les interrupcions es la següent: Mòdul 2 (als 5 ms), Mòdul 3 (als 12 ms) i Mòdul 1 (als 20 ms). Considerar també que quan s'atén una interrupció el sistema d'interrupcions queda inhibit.

No apropiatiu



Apropiatiu



Problema 11.

Per transmetre **rafegues** de fins a **64 bytes**, s'ha afegit un buffer de 64 bytes al controlador de DMA. Cada cop que el controlador DMA demana el bus a la CPU, aquesta li dona prioritat immediatament. Per a la transferència d'**una ràfega o bloc de dades de 64 bytes (de 4 bytes en 4 bytes, pel màxim ample de banda del bus de dades)** entre el controlador de disc i la memòria, s'utilitza el bus de memòria durant **18 ns** i es torna a cedir l'ús del bus a la CPU. Aquests 18 ns inclouen el temps de cessió del bus (1 ns) , el temps de la transferència pel bus (1 ns per cada dada de 4 bytes), i el temps de recuperació del bus (1 ns). La programació i la finalització de la transmissió per DMA (amb una RSI) consumeixen un total (entre les dues tasques) de 1 us de temps d'execució de la CPU.

a) Suposant que la CPU no pot fer cap tasca durant tot el temps en que el bus està ocupat per part del controlador DMA. Per la transferència de l'exercici 6, de 8 Kbytes entre el controlador del disc dur i un buffer de memòria. Quin percentatge de temps està ocupada la CPU sense poder executar codi efectiu d'altres programes durant la transferència per DMA amb rafegues de 64 bytes?

- Número de peticiones al Bus:

2^{13} (8 Kbytes) / 2^6 (bytes por ráfaga) = 2^7 o 128 ráfagas de 64 bytes

- Tiempo total de ocupación del Bus:

128 ráfagas * 18 ns por ráfaga = 2304 ns o 2,3 us.

- Tiempo ejecución CPU dedicado a la transferencia (programación y finalización DMA + tiempo ocupación del bus por el DMA):

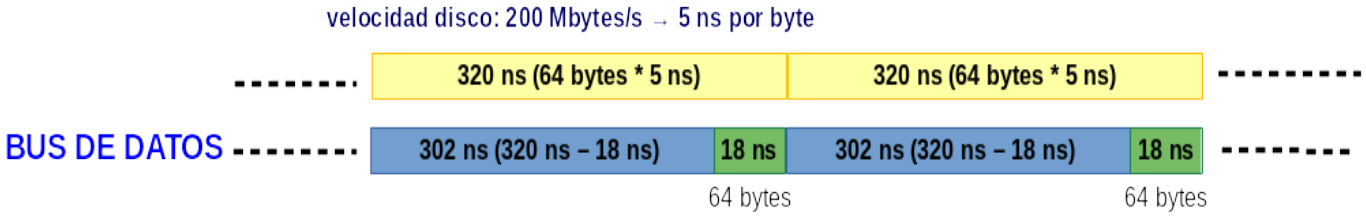
1 us (programación y finalización DMA) + 2,3 us =3 ,3 us

- Tiempo total transferencia respecto CPU (programación y finalización DMA + tiempo transferencia del disco): 40 us

- Porcentaje de tiempo ocupado:

(100 * 3,3 us) / 40 us = 8,25 %

(En el problema 9 , DMA byte a byte, % tiempo ejecución CPU ocupada: 64%)



Tiempo para la lectura de 64 bytes del disco duro al buffer del DMA (incluye los comandos necesarios para esta operación). En los últimos 18 ns se transmiten los datos del buffer del DMA a la memoria.

Tiempo del bus de datos ocupado por la ráfaga del DMA (transmitir 64 bytes del buffer del DMA al buffer de la memoria).

Tiempo del bus que puede utilizar la CPU para ejecutar instrucciones (mientras se leen los 64 bytes del disco duro y se sitúan en el buffer del DMA).

b) Les mateixes condicions que l'apartat anterior, però la CPU pot seguir executant instruccions durant el 25% del temps en que el bus està ocupat per part del controlador DMA. Quin percentatge de temps està ocupada la CPU sense poder executar codi efectiu d'altres programes durant la transferència?

- Tiempo ejecución CPU dedicado a la transferencia (programación y finalización DMA + 75% tiempo ocupación del bus por el DMA):

1 us + (2,3 *75)/100 us =1 us + 1,7 us = 2,7 us

- Porcentaje de tiempo ocupado:

(100 * 2,7 us) / 40 us =6,75 %