Introducció als Computadors

Tema 11: Memòria

http://personals.ac.upc.edu/enricm/Docencia/IC/IC11.pdf

Enric Morancho
(enricm@ac.upc.edu)

Departament d'Arquitectura de Computadors
Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

2020-21, 1^{er} quad.

Publicitat (setembre 1977)

[1]

- Dissenyarem un sistema de memòria que podrà treballar tant a nivell de *byte* com a nivell de *word* (2 *bytes*)

Índex

Full de ruta

	Tema							
	7	8	9	10	11	12	13	14
Unitat de Control	UCE	UCE	UCE	UCG	UCG	UCG	UCG	UCG
Unitat de Procés	UPE	UPG	UPG	UPG	UPG	UPG	UPG	UPG
Entrada/Sortida	-	-	IO	IO	IO	IO	IO	IO
Memòria RAM	-	-	-	-	MEM	MEM	MEM	MEM
Harvard unicicle	-	-	-	-	-	✓	-	-
Harvard multicicle	-	-	-	-	-	✓	-	-
Von Neumann	-	-	-	-	-	-	✓	✓
Lleng. *assembler*	-	-	-	✓	✓	✓	✓	✓

Escenari

- La majoria de programes "útils" necessiten manipular molta més informació de la que hi cap a 8 registres de 16 bits (128 bits)
- Augmentar el nombre de registres no és viable
  - A la codificació SISA no hi ha espai per a identificar més registres
  - Podria impactar en el T_c
    - Tema 12 :-)

Objectiu del tema

- Afegir un mòdul de memòria RAM al nostre computador
  - RAM = *Random Access Memory*
    - Volàtil
    - Hi podrem emmagatzemar "molta" informació amb un cost reduït
    - És més lenta que el banc de registres
  - La RAM té un seguit de posicions
    - Cada posició s'identifica amb una adreça (un natural entre 0 i $2^n - 1$)
    - A cada posició s'hi emmagatzema el mateix nombre de bits (8, un *byte*)
    - El temps d'accés és independent de la posició accedida
  - S'hi podran fer lectures i escriptures de *byte* y de *word* (16 bits)
  - Asíncron
- En aquest tema veurem:
  - Interfície del mòdul de memòria
  - Ampliació del LM SISA per a accedir al mòdul
  - Connexió del mòdul amb la UPG i ampliació de la Lògica de Control
  - Implementació del mòdul de memòria utilitzant mòduls de memòria d'amplada 1 *byte*

Índex

Mòdul de memòria RAM

- Definicions:
  - 1 *byte* = 8 bits
  - 1 *word* = amplada dels registres del processador
    - Al nostre cas, 1 *word* = 2 *bytes* = 16 bits
- Voldrem poder operar amb dades de tipus *byte* i de tipus *word*
  - Típicament, els caràcters es codifiquen amb un *byte*
  - Depenent del rang de valors a representar, un natural/enter es podrà codificar amb un *byte* o amb un *word*
  - Per compatibilitat, els processadors Intel/AMD de 64 bits permeten operar amb dades de 8, 16, 32 i 64 bits
- **El mòdul haurà de permetre una lectura/escriptura a *byte*/*word* cada cicle de la UPG**
- El construirem amb mida 2^{16} *bytes*, equivalent a 2^{15} *words*
  - MEM-64KB-32KW
- Comunicació **asíncrona** amb la UPG

Mòdul de memòria: adreçament

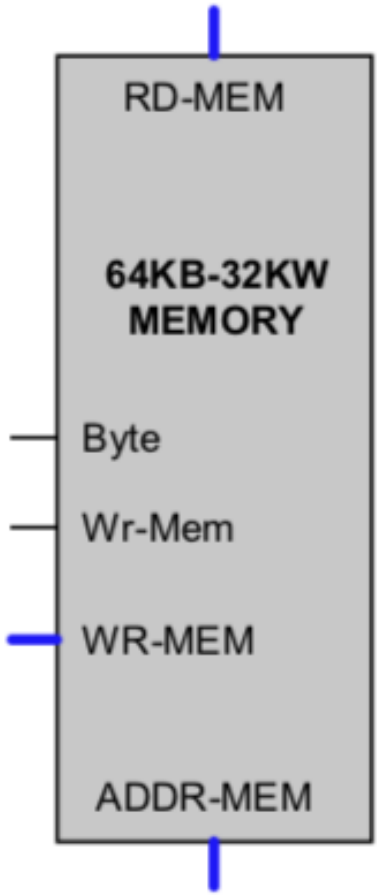
- Cada posició del mòdul de memòria té una adreça (@) de 16 bits
  - Des de l'adreça 0x0000 fins a la 0xFFFF
  - 2^{16} adreces diferents
- Adreçament a nivell de *byte*
  - El mòdul emmagatzema 2^{16} *bytes*
    - Des de l'adreça 0x0000 a la 0xFFFF
- Adreçament a nivell de *word*
  - El mòdul emmagatzema 2^{15} *words*
    - Adreces "parells" 0x0000, 0x0002, ... 0xFFFE
  - El *word* està format per dos *bytes* amb adreces consecutives
    - Els indicats per l'adreça i per adreça + 1
    - El *byte* de menys pes del *word* és el l'adreça "parell"
    - Emmagatzemament *Little endian*
  - Són decisions de disseny, podríem haver-ne pres d'altres
    - Permetre *words* desalineats
    - Emmagatzemament *Big endian*

Mòdul de memòria: exemple dues visions

byte addressing		*word* addressing	
@	MEM_b[@]	MEM_w[@]	@
0x0000	0x7C	0xB57C	0x0000
0x0001	0xB5		
0x0002	0xDC	0xACDC	0x0002
0x0003	0xAC		
0x0004	0xFE	0xCAFE	0x0004
0x0005	0xCA		
	...	...	
0x124E	0x1A	0xD21A	0x124E
0x124F	0xD2		
	...	...	
0xFFFE	0xBE	0xBABE	0xFFFE
0xFFFF	0xBA		

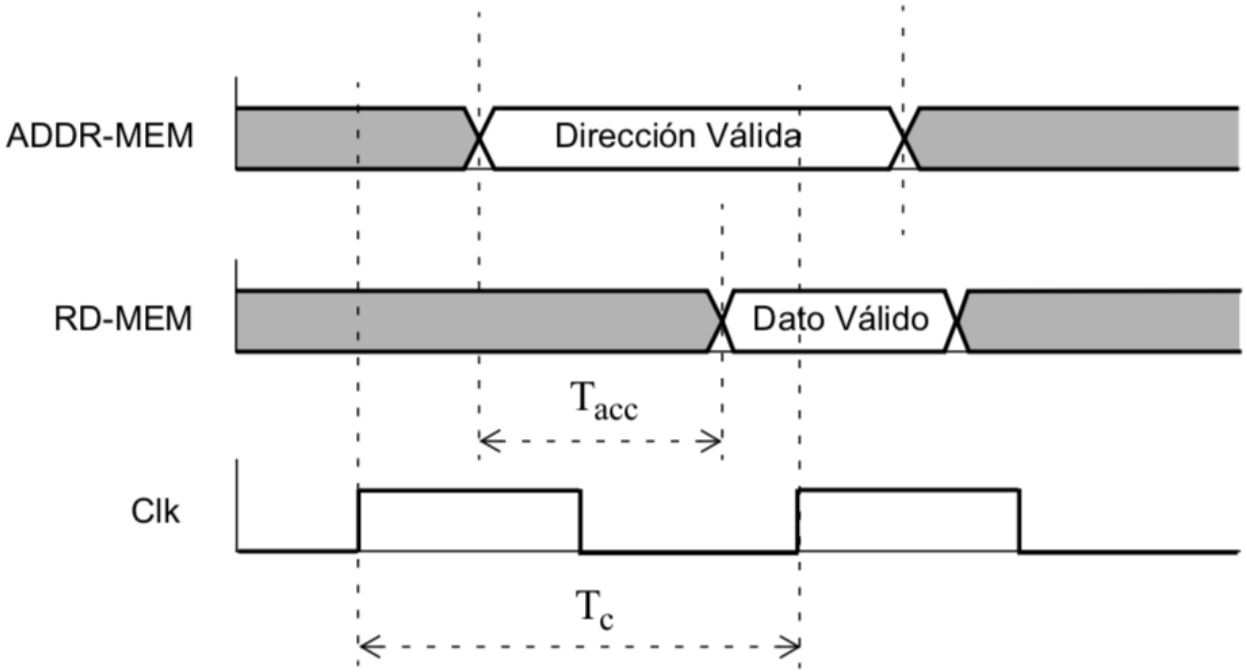
Encapsulat del mòdul MEM-64KB-32KW

- Entrades:
  - ADDR-MEM: bus de 16 bits amb l'adreça a accedir
  - Byte: senyal d'un bit que indica que si l'accés és a *byte* ("1") o a *word* ("0")
  - Wr-Mem: senyal d'un bit que indica si es fa una escriptura ("1") o una lectura ("0")
  - WR-MEM: bus de 16 bits amb la dada a escriure
- Sortides:
  - RD-MEM: bus de 16 bits amb la dada llegida
- Compte amb la temporalitat!
  - A les escriptures, Wr-Mem hauria de passar a "1" quan els busos ADDR-MEM i WR-MEM tinguin un valor estable



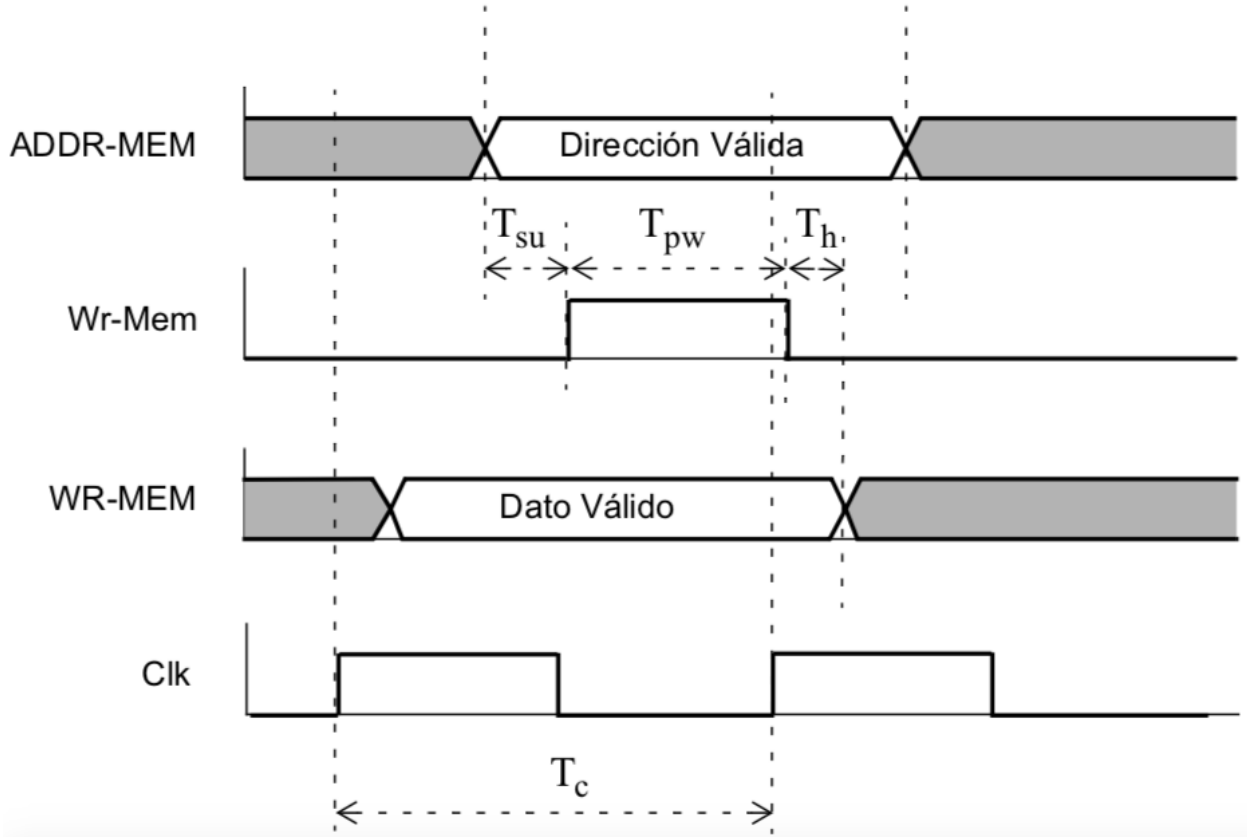
Cronograma lectura

- Un cop que ADDR-MEM és estable, la dada vàlida estarà disponible un cop transcorregudes T_{acc} unitats de temps



Cronograma escriptura

- Al tema 12 garantirem la correcta temporalitat dels senyals!
  1. Cal estabilitzar WR-MEM, ADDR-MEM i Byte (no mostrat al crono.)
  2. Cal esperar T_{su} u.t. abans de posar Wr-Mem a "1" (T_{setup})
  3. Cal mantenir Wr-Mem a "1" per T_{pw} u.t. ($T_{pulse\ width}$)
  4. Després de pasar Wr-Mem a "0", cal continuar mantenint estables WR-MEM i ADDR-MEM T_h u.t. (T_{hold})

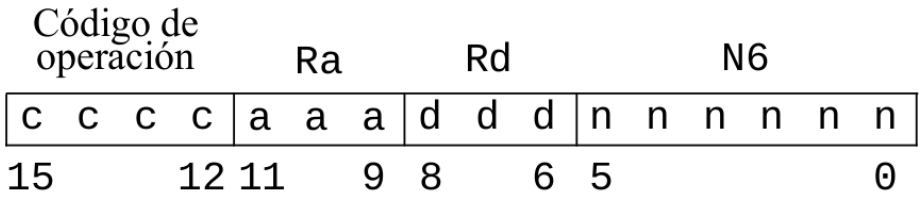


Índex

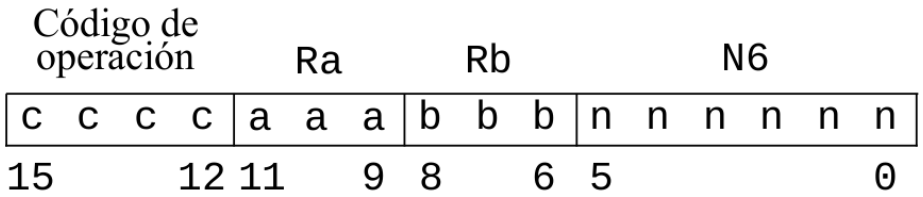
Noves instruccions SISA: *Load*, *Store*

- Afegim quatre noves instruccions al LM SISA
  - `LD`, `LDB`, `ST`, `STB`
    - 2 per llegir (`LD`, `LDB`), 2 per escriure (`ST`, `STB`)
    - 2 per accedir a *byte* (`LDB`, `STB`), 2 per accedir a *word* (`LD`, `ST`)
- Totes utilitzaran la codificació 2-R

LD y LDB



ST y STB



- `Ra` i `N6` determinaran l'adreça a accedir
- Cada instrucció tindrà un codi d'operació diferent

Sintaxi/Semàntica noves instruccions

- `LDB Rd, N6(Ra)`
  - $\texttt{Rd} \leftarrow \texttt{SE}(\texttt{MEM}_\texttt{b}[\texttt{Ra} + \texttt{SE}(\texttt{N6})]);$ $\texttt{PC} \leftarrow \texttt{PC} + 2;$
- `STB N6(Ra), Rb`
  - $\texttt{MEM}_\texttt{b}[\texttt{Ra} + \texttt{SE}(\texttt{N6})] \leftarrow \texttt{Rb} < 7..0 >;$ $\texttt{PC} \leftarrow \texttt{PC} + 2;$
    - Només s'emmagatzema a memòria els *byte* baix de `Rb`
- `LD Rd, N6(Ra)`
  - $\texttt{Rd} \leftarrow \texttt{MEM}_\texttt{w}[(\texttt{Ra} + \texttt{SE}(\texttt{N6}))\& \sim 1];$ $\texttt{PC} \leftarrow \texttt{PC} + 2;$
    - $\sim 1 = 0xFFFE$
    - & és l'operació AND bit a bit
    - El bit de menys pes del resultat de la suma es posa a 0 perquè, com accedim a *word*, l'adreça ha de ser un nombre parell
- `ST N6(Ra), Rb`
  - $\texttt{MEM}_\texttt{w}[(\texttt{Ra} + \texttt{SE}(\texttt{N6}))\& \sim 1] \leftarrow \texttt{Rb};$ $\texttt{PC} \leftarrow \texttt{PC} + 2;$

Codificació SISA

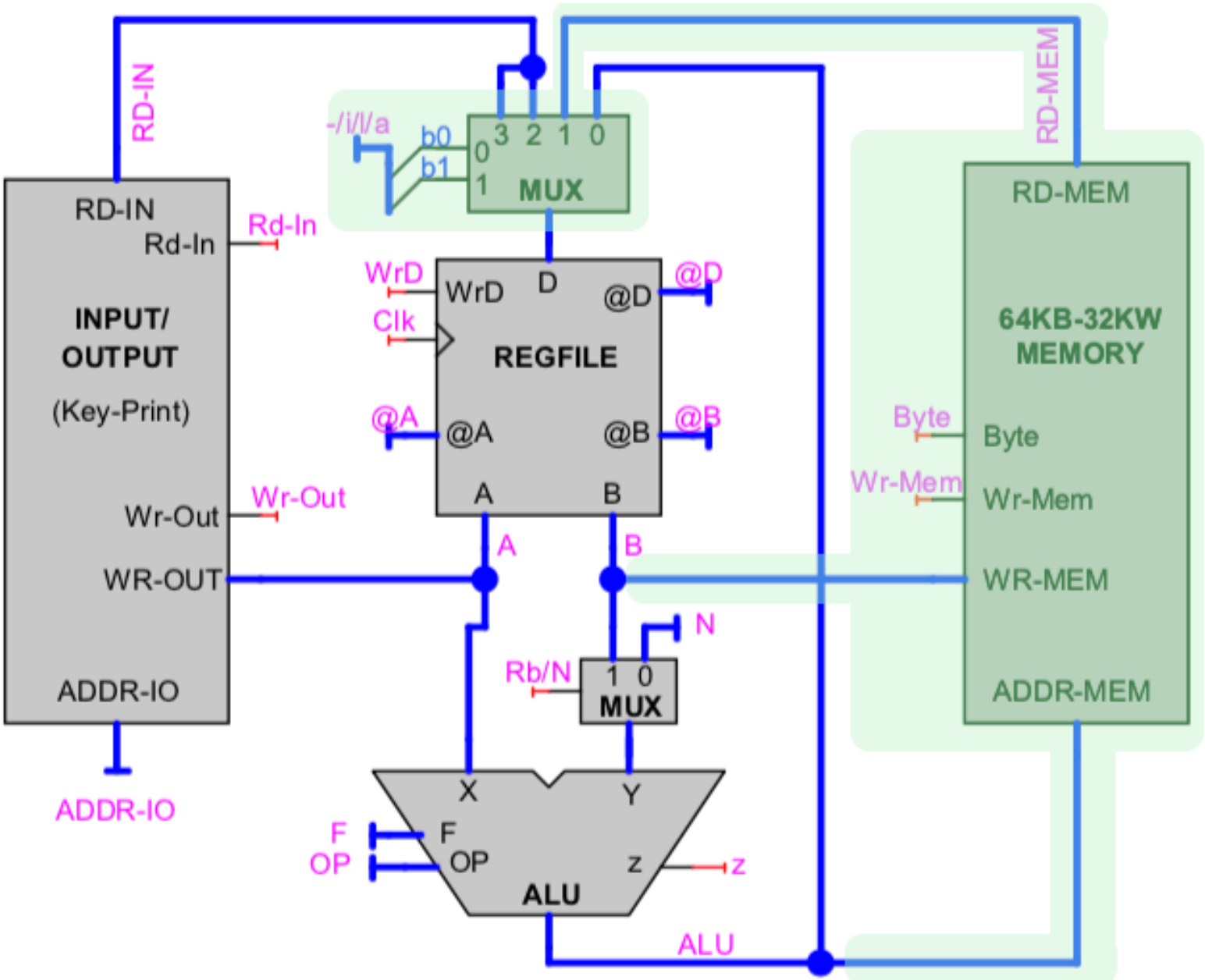
15 14 13 12	11 10 9	8	7 6	5 4 3	2 1 0	Name	Mnemonic	Format
0 0 0 0	a a a	b	b b	d d d	f f f	Logic and Aritmetic Operations	AND, OR, XOR, NOT, ADD, SUB, SHA, SHL	3R
0 0 0 1	a a a	b	b b	d d d	f f f	Compare Signed and Unsigned	CMPLT, CMPLE, -, CMPEQ, CMPLTU, CMPLEU, -, -	
0 0 1 0	a a a	d	d d	n n n	n n n	Add Immediate	ADDI	
0 0 1 1	a a a	d	d d	n n n	n n n	Load	LD	2R
0 1 0 0	a a a	b	b b	n n n	n n n	Store	ST	
0 1 0 1	a a a	d	d d	n n n	n n n	Load Byte	LDB	
0 1 1 0	a a a	b	b b	n n n	n n n	Store Byte	STB	
0 1 1 1							Branch future extension	
1 0 0 0	a a a	0	n n n n n n n			Branch on Zero	BZ	1R
		1				Branch on Not Zero	BNZ	
1 0 0 1	d d d	0	n n n n n n n			Move Immediate	MOVI	
	a a a	1				Move Immediate High	MOVHI	
	d d d							
1 0 1 0	d d d	0	n n n n n n n			Input	IN	
	a a a	1				Output	OUT	
1 0 1 1	x x x x x x x x x x x x						Future extensions	
1 1 x x								

Índex

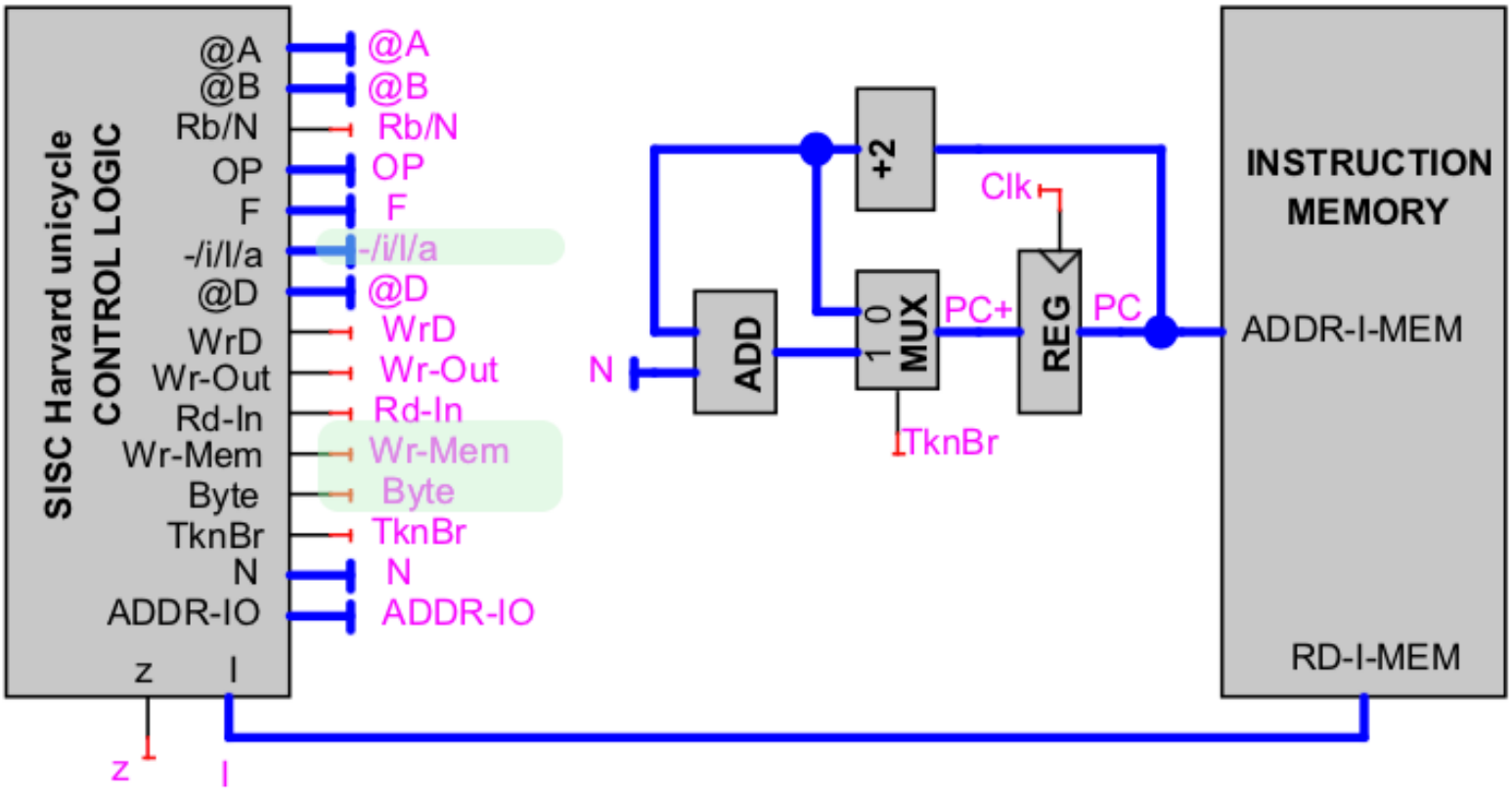
Connectem el mòdul de memòria a la UPG

- L'adreça ADDR-MEM serà la sortida de la ALU
  - Així podrem sumar `Ra` i `N6` com a les instruccions `ADDI`
- La sortida RD-MEM l'encaminarem al banc de registres
  - Però el multiplexor MUX 2-1 In/Alu no té cap entrada lliure!
  - El substituirem per un MUX 4-1
    - Tindrà dos senyals de selecció
- El bus WR-MEM s'alimentarà des de `Rb` del banc de registres
- La resta de senyals les generarà la lògica de control
  - Byte, Wr-Mem

Esquema UPG + IO + MEM

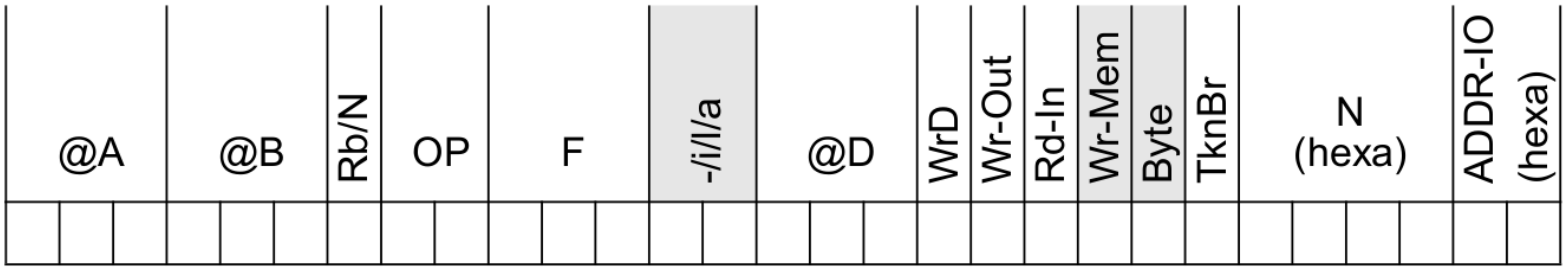


Esquema Lògica de control i seqüenciament



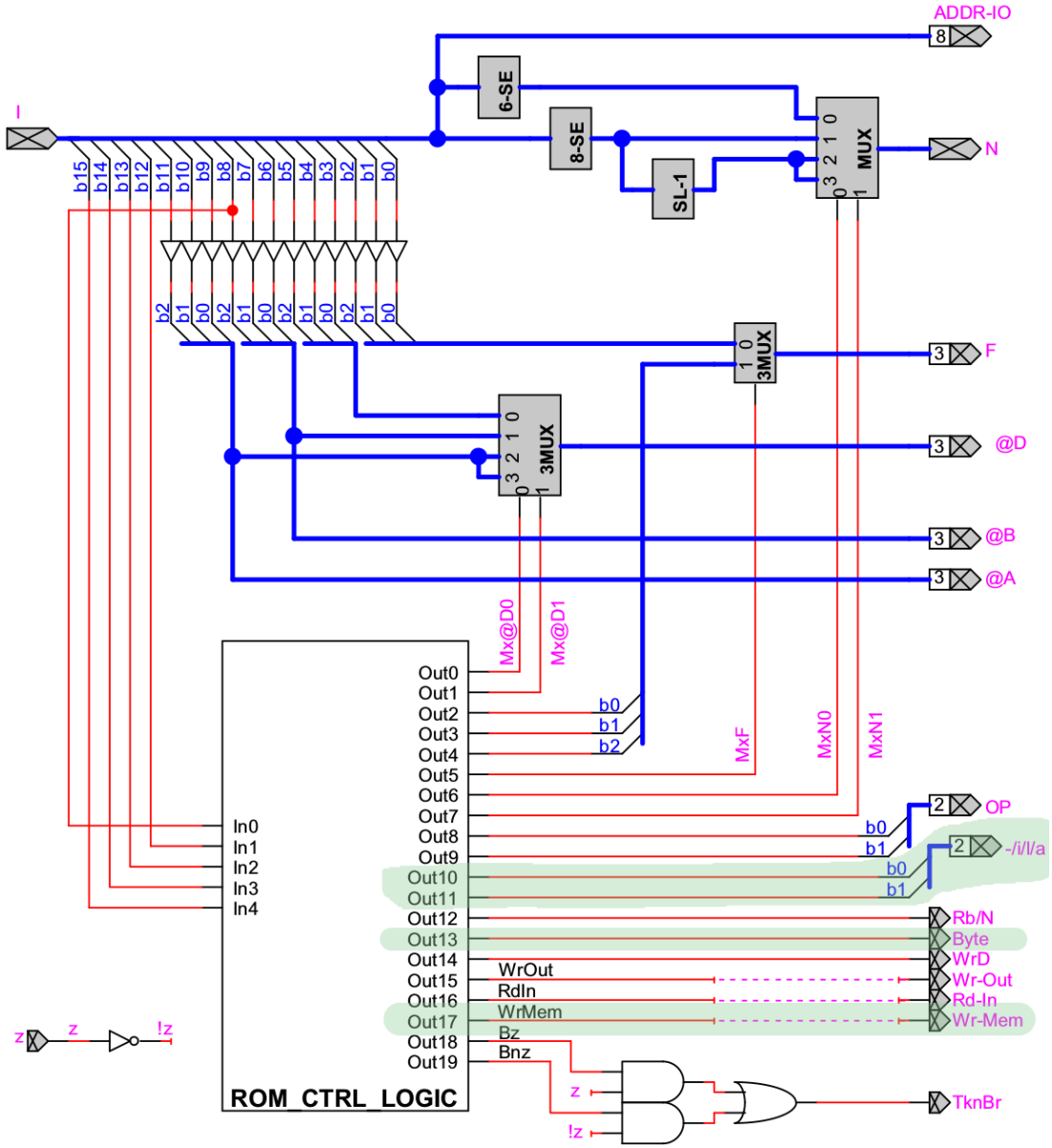
Lògica de control: paraula de control

- Passarà a tenir 47 bits
  - Afegim senyals Wr-Mem i Byte
    - Wr-Mem mai podrà valer `x`, valdrà "1" per `ST*`, "0" per a la resta
    - Byte valdrà "1" per `STB` i `LDB`, "0" per `ST` i `LD`, `x` per a la resta
  - Canviem senyal In/Alu (1 bit) per bus -/i/l/a (2 bits)



- Els nous senyals els generarà la ROM_CTRL_LOGIC a partir del codi d'operació de la instrucció en execució

Lògica de control: implementació



Lògica de control: ROM_CTRL_LOGIC

Dirección					Contenido																			
In4	In3	In2	In1	In0	Out19	Out18	Out17	Out16	Out15	Out14	Out13	Out12	Out11	Out10	Out9	Out8	Out7	Out6	Out5	Out4	Out3	Out2	Out1	Out0	
I<15>	**I<14>**	**I<13>**	**I<12>**	**I<8>**	Bnz	Bz	Wr-Mem	Rdln	WrOut	WrD	Byte	Rb/N	-/i/l/a1	-/i/l/a0	OP1	OP0	MxN1	MxN0	MxF	F2	F1	F0	Mx@D1	Mx@D0	
0	0	0	0	x	0	0	0	0	0	1	x	1	0	0	0	0	x	x	0	x	x	x	0	0	AL
0	0	0	1	x	0	0	0	0	0	1	x	1	0	0	0	1	x	x	0	x	x	x	0	0	CMP
0	0	1	0	x	0	0	0	0	0	1	x	0	0	0	0	0	0	0	1	1	0	0	0	1	ADDI
0	0	1	1	x	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0	1	LD
0	1	0	0	x	0	0	1	0	0	0	0	0	x	x	0	0	0	0	1	1	0	0	x	x	ST
0	1	0	1	x	0	0	0	0	0	1	1	0	0	1	0	0	0	0	1	1	0	0	0	1	LDB
0	1	1	0	x	0	0	1	0	0	0	1	0	x	x	0	0	0	0	1	1	0	0	x	x	STB
0	1	1	1	x	0	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	(NOP)
1	0	0	0	0	0	1	0	0	0	0	x	x	x	x	1	0	1	0	1	0	0	0	x	x	BZ
1	0	0	0	1	1	0	0	0	0	0	x	x	x	x	1	0	1	0	1	0	0	0	x	x	BNZ
1	0	0	1	0	0	0	0	0	0	1	x	0	0	0	1	0	0	1	1	0	0	1	1	0	MOVI
1	0	0	1	1	0	0	0	0	0	1	x	0	0	0	1	0	0	1	1	0	1	0	1	0	MOVHI
1	0	1	0	0	0	0	0	1	0	1	x	x	1	0	x	x	x	x	x	x	x	x	1	0	IN
1	0	1	0	1	0	0	0	0	1	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	OUT
1	0	1	1	x	0	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	(NOP)
1	1	x	x	x	0	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	(NOP)

Computador SISC-Harvard unicicle

- Hem completat el disseny del nostre primer computador
  - **Computador Harvard unicicle**
    - Harvard: memòries de dades i d'instruccions independents
    - Unicicle: Totes les instruccions triguen un cicle en executar-se
- Al tema següent determinarem el temps de cicle T_c del computador
  - Constatarem que algunes instruccions triguem més que d'altres
    - Instruccions "lentes" (accessos a memòria) vs. "ràpides" (les demés)
  - Amb una petita modificació a la lògica de control, farem una versió "multicicle" del computador Harvard
    - Redefinirem el temps de cicle
    - Les instruccions "lentes" trigaran 4 cicles i les "ràpides" 3

Índex

Enunciat

- Fer un programa en LM SISA que calculi l'histograma d'un conjunt de 1.024 dades
- Cada dada és un vector X de 16 bits però ens garanteixen el rang de valors $0 \leq X_u \leq 255$
- Les dades entren des del teclat
- El programa ha de comptar quantes dades valen 0, quantes valen 1, ..., quantes valen 255
- Un cop processades totes les dades, cal mostrar l'histograma pel dispositiu impressora
  - Començant pel nombre de repeticions del valor "0"i acabant pel nombre de repeticions del valor "255"

Anàlisi: necessitat de memòria

- L'histograma necessita 256 comptadors
- En el cas extrem, un comptador hauria de podem comptar des de 0 fins a 1.024
  - Per evitar overflows, el comptador hauria de tenir un mínim de $\lceil log_2 1025 \rceil = 11$ bits
  - Com el computador és de 16 bits, cada comptador ocuparà un *word* (16 bits)
- Utilitzarem un vector HISTO amb 256 *words*
- Com no hi cap al REGFILE, el guardarem a la RAM
- Assumirem que l'adreça inicial del vector HISTO és 0x3AF0
  - HISTO[0] s'emmagatzema a 0x3AF0 i 0x3AF1
  - HISTO[1] s'emmagatzema a 0x3AF2 i 0x3AF3
  - ...
  - HISTO[255] s'emmagatzema a 0x3CEE i 0x3CEF

Solució: Fases

- Dividirem el programa en tres fases
  1. Inicialització
  2. Entrada de dades i càlcul de l'histograma
  3. Presentació del resultat

Fase 1: Inicialització

- Cal inicialitzar totes les posicions del vector HISTO a 0x0000
- Bucle de 256 iteracions
- A cada iteració
  - Escriure el *word* 0x0000 a una posició del vector
- Utilitzarem dos registres
  - Un per a controlar el nombre d'iteracions pendents del bucle
    - S'inicialitza a 256
    - A cada iteració es decrementarà en 1
  - Un altre per saber a quina adreça de memòria s'ha d'escriure el *word*
    - S'initicalitzarà amb l'adreça inicial del vector
    - A cada iteració s'incrementarà en 2 perquè cada *word* ocupa dos *bytes*

Fase 1: Inicialització


```
;***** Inicialización
      MOVI  R1, 0xF0
      MOVHI R1, 0x3A    ;R1 apunta a HISTO[0]
      MOVI  R7, 0x00    ;R7 = 0x0000 (Ext. signo)
      MOVHI R7, 0x01    ;R7 = 0x0100 (cte. 256 en decimal)
      MOVI  R0, 0       ;constante 0x0000
Loop1: ST   0(R1), R0   ;pone a 0x0000 un elemento de HISTO
      ADDI  R1, R1, 2   ;incrementa puntero
      ADDI  R7, R7, -1  ;decrementa contador iter. restantes
      BNZ   R7, -4      ;¿otra iteracion?
```

Fase 2: Entrada dades i càlcul histograma

- Bucle de 1.024 iteracions
- A cada iteració
  - Llegir una dada de teclat
    - Cal espera activa fins que la dada estigui disponible
  - Incrementar el comptador corresponent a la dada llegida
    - Cal calcular a quina adreça de memòria es troba
    - Carregar-lo a un registre
    - Incrementar el registre
    - Escriure'l a memòria
  - Decrementar el comptador d'iteracions
    - Inicialitzat a 1.024

Fase 2: Entrada dades i càlcul histograma


```
;***** Entrada de operandos y cálculo del histograma
      MOVI  R6, 0xF0
      MOVHI R6, 0X3A    ;R6, constante HISTO
      MOVI  R7, 0x00
      MOVHI R7, 0x04    ;R7, contador iter. restantes = 1024
Loop2: IN   R1, KEY-STATUS
      BZ    R1, -2
      IN    R1, KEY-DATA  ;entra Xu del teclado
      ADD   R1, R1, R1    ;multiplica R1 por 2
      ADD   R1, R1, R6    ;calcula dirección de HISTO[Xu]
      LD    R5, 0(R1)     ;lee HISTO[Xu]
      ADDI  R5, R5, 1     ;incrementa valor
      ST    0(R1), R5     ;actualiza HISTO[Xu]
      ADDI  R7, R7, -1    ;decrementa contador iteraciones
      BNZ   R7, -10       ;¿otra iteración?
```

Fase 3: Presentació resultats

- Bucle de 256 iteracions
- A cada iteració
  - Llegir un valor del histograma
  - Mostrar-lo per impressora
    - Cal espera activa per esperar disponibilitat
  - Decrementar el comptador d'iteracions
    - Inicialitzat a 256

Fase 3: Presentació resultats


```
;***** Salida de resultados
      MOVI  R7, 0x00
      MOVHI R7, 0x01 ;R7, contador iter. restantes a 256
Loop3: IN    R0, PRINT-STATUS
      BZ    R0, -2
      LD    R0, 0(R6)
      OUT   PRINT-DATA, R0 ;imprime elemento del histograma
      ADDI  R6, R6, 2      ;incrementa puntero a HISTO
      ADDI  R7, R7, -1 ;decrementa contador iter. restantes
      BNZ   R7, -7         ;¿otra iteración?
```

Índex

Implementació mòdul de memòria

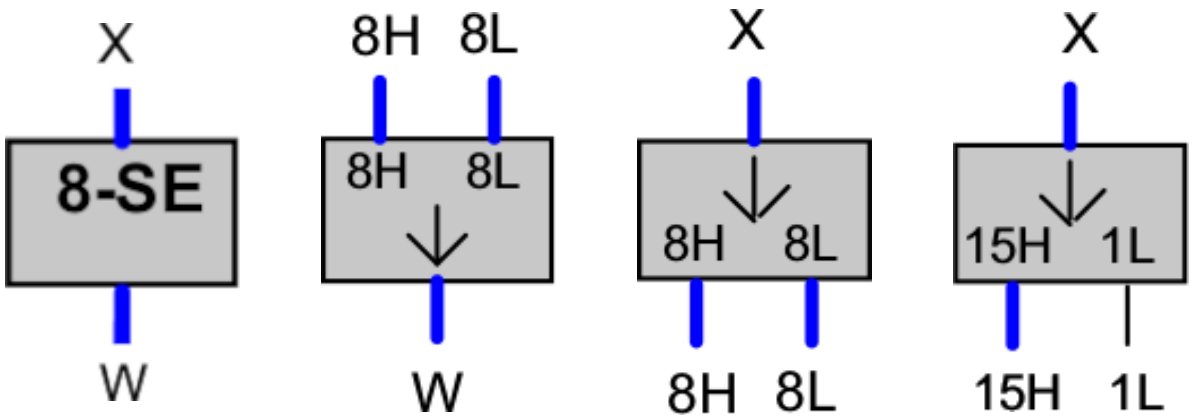
- Farem **una possible** implementació del mòdul de memòria descrit
- Hipòtesis:
  - El fabricant proporciona mòduls de memòria d'amplada *byte*
  - A cada cicle de la UPG només podrem llegir un *byte* del mòdul
- Com llegirem un *word* (2 *bytes*) en un cicle?
  - Caldrà tenir dos mòduls de memòria treballant a la vegada
  - Com distribuirem (*mapejarem*) els *bytes* entre els dos mòduls?

Distribució de les dades

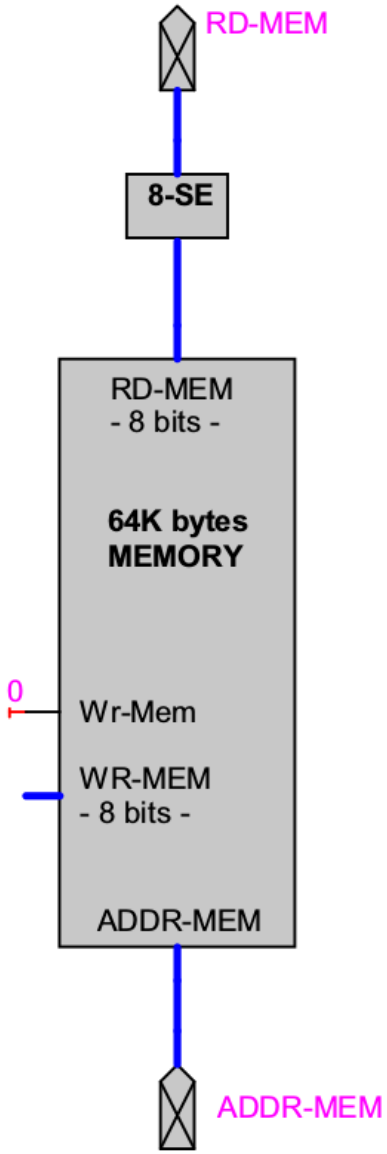
- Haurem de distribuir els 2^{16} *bytes* entre dos mòduls de 2^{15} *bytes*
- Volem poder accedir als dos *bytes* d'un *word* en un únic cicle
  - Això obliga a que adreces de 16 bits consecutives no vagin a parar al mateix mòdul
  - Distribució *entrellaçada* (*interleaved*)
- Anomenarem els mòduls Mòdul1 i Mòdul0
  - El Mòdul0 emmagatzemarà les adreces "parells"
  - El Mòdul1 emmagatzemarà les adreces "senars"
- Caldrà traduir les adreces de 16 bits a adreces de 15 bits
  - Els mòduls de 2^{15} *bytes* tenen 15 bits d'adreça!

Implementació: CLC's auxiliars

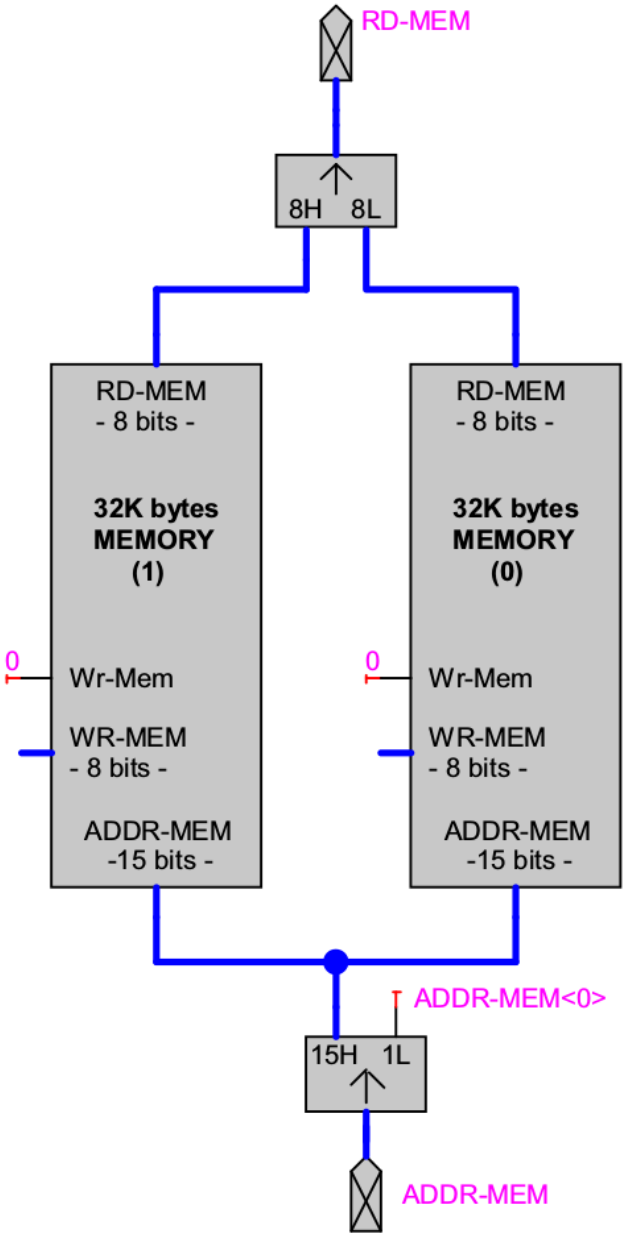
- Estendre el bit de signe a un bus de 8 bits fins a 16 bits
- Combinar dos busos de 8 bits en un bus de 16 bits
- Separar un bus de 16 bits en dos busos de 8 bits
- Separar un bus de 16 bits en dues parts:
  - Un bus de 15 bits (els 15 bits alts)
  - Un bit (el bit baix)



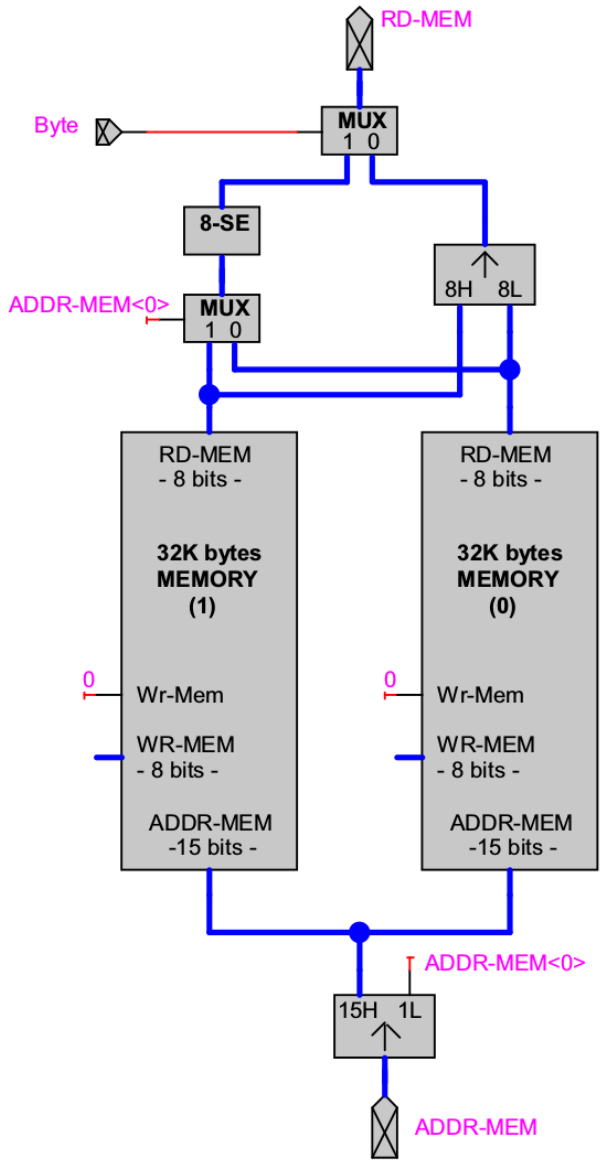
Implementació lectura: només *byte*



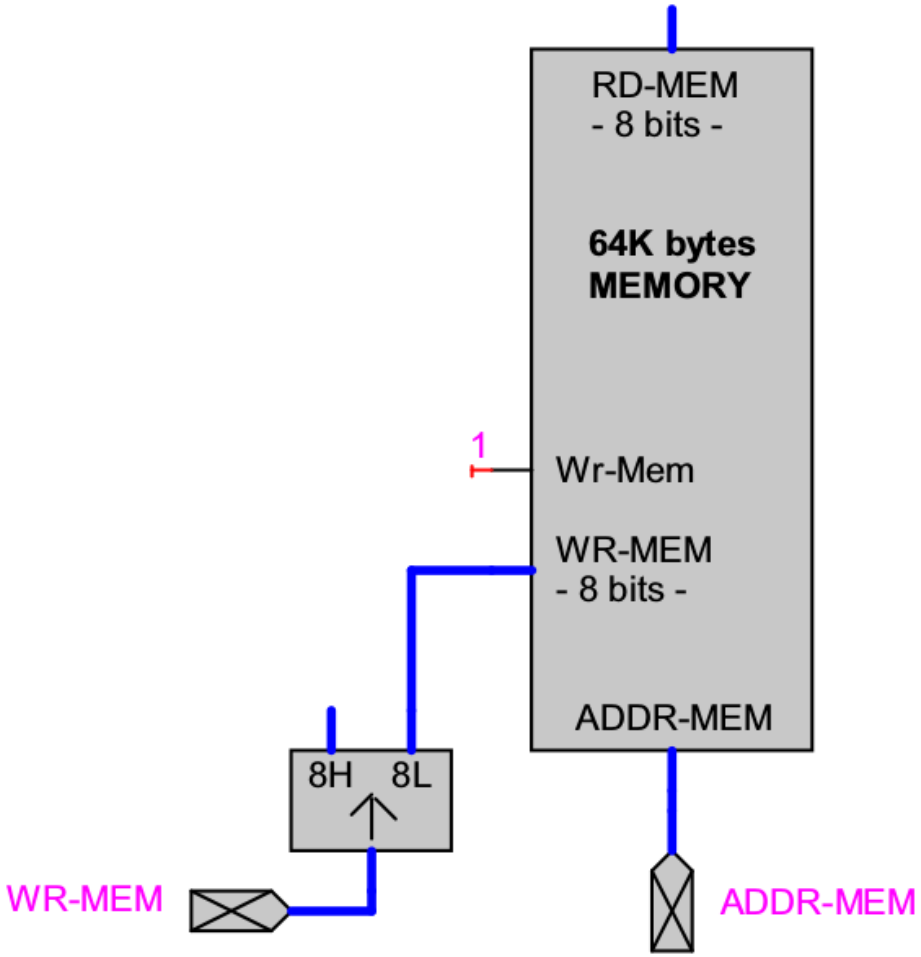
Implementació lectura: només *word*



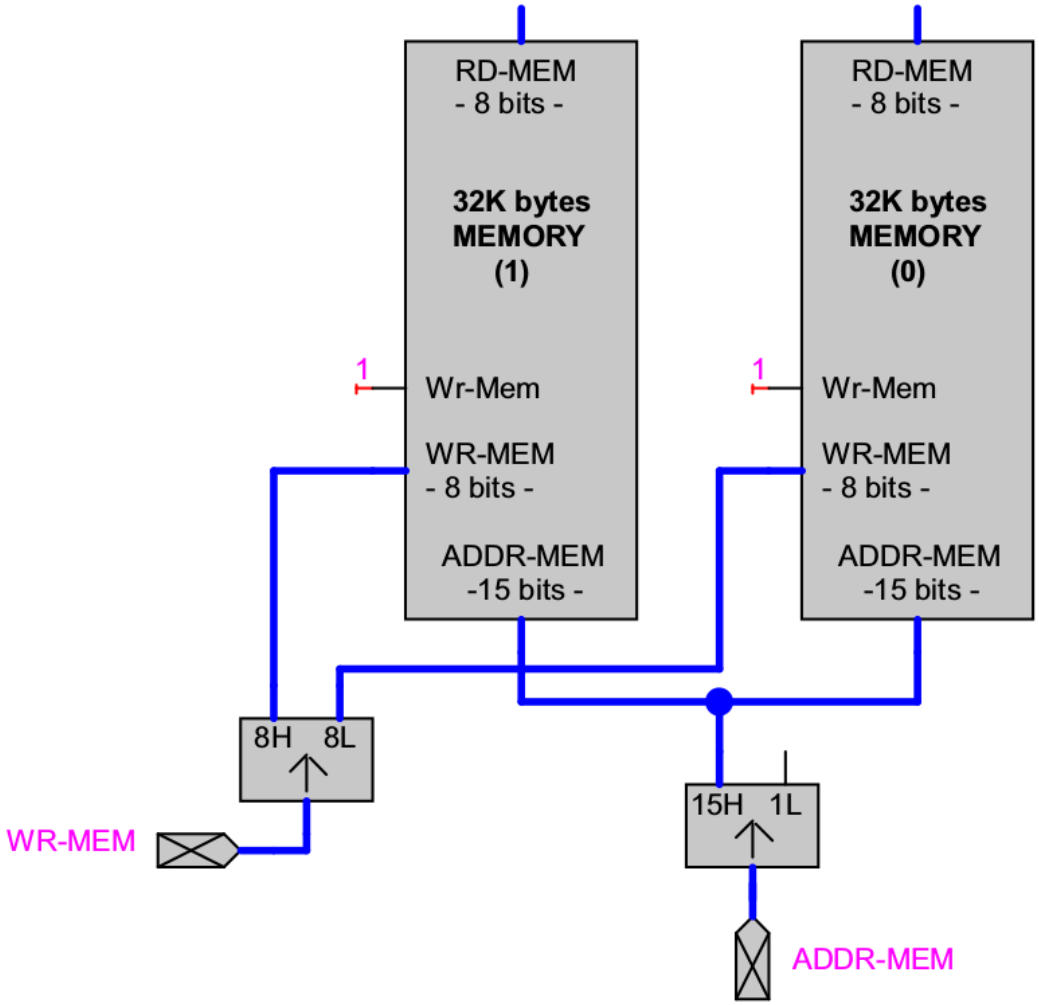
Implementació lectura *byte* o *word*



Implementació escriptura: només *byte*

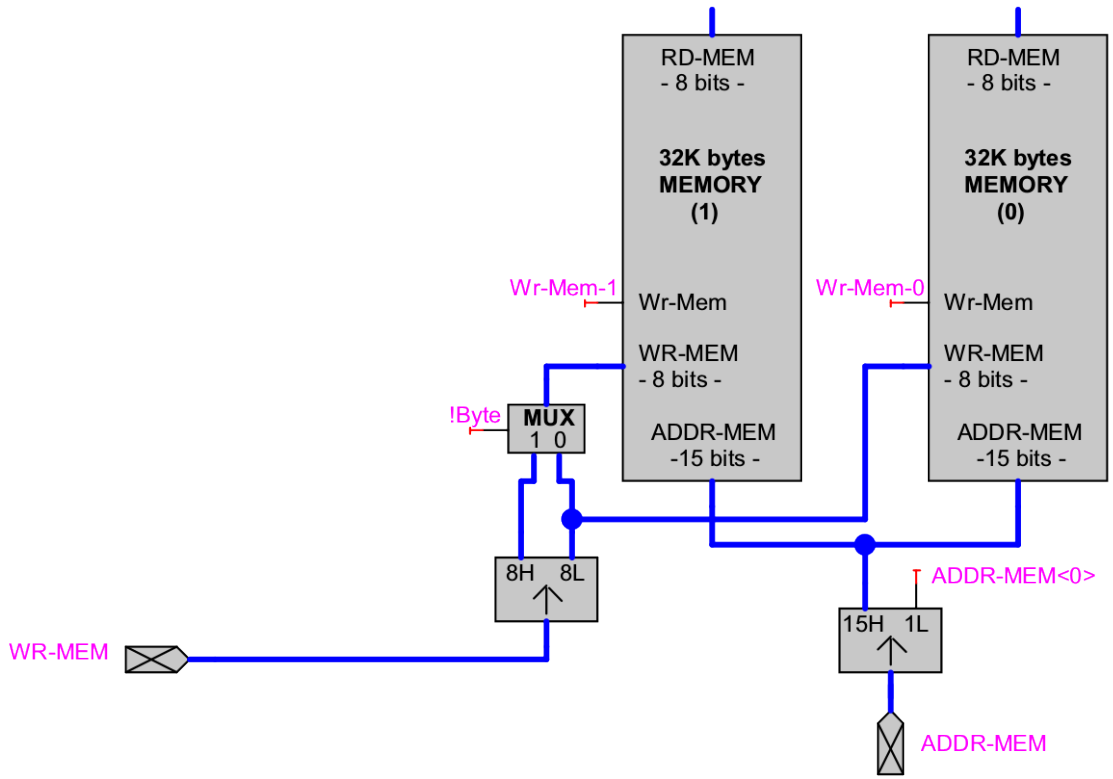


Implementació escriptura: només *word*

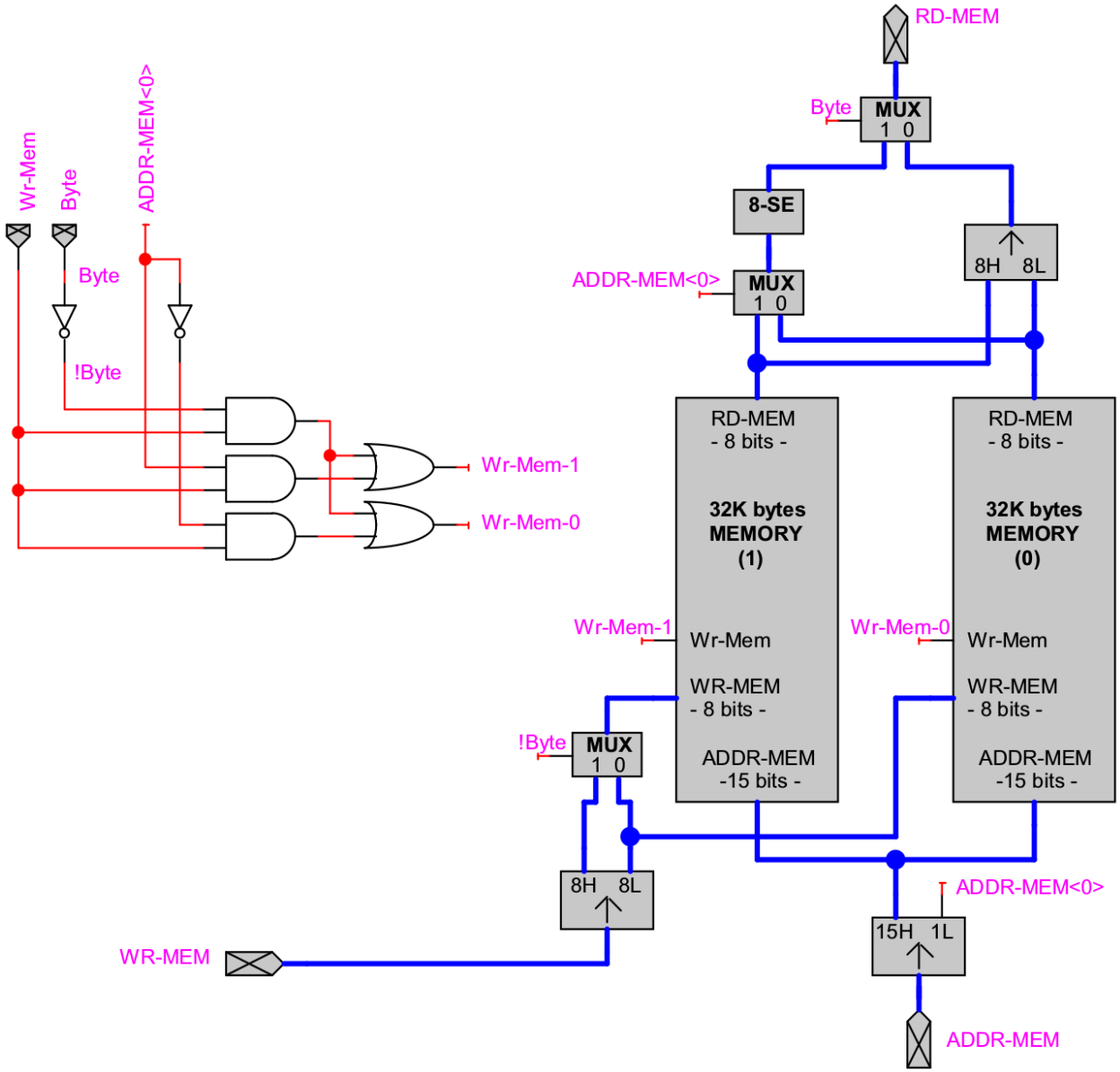


Implementació escriptura *byte* o *word*

Wr-Mem	Byte	ADDR-MEM$_0$	Wr-Mem$_1$	Wr-Mem$_0$
0	x	x	0	0
1	0	x	1	1
1	1	0	0	1
1	1	1	1	0

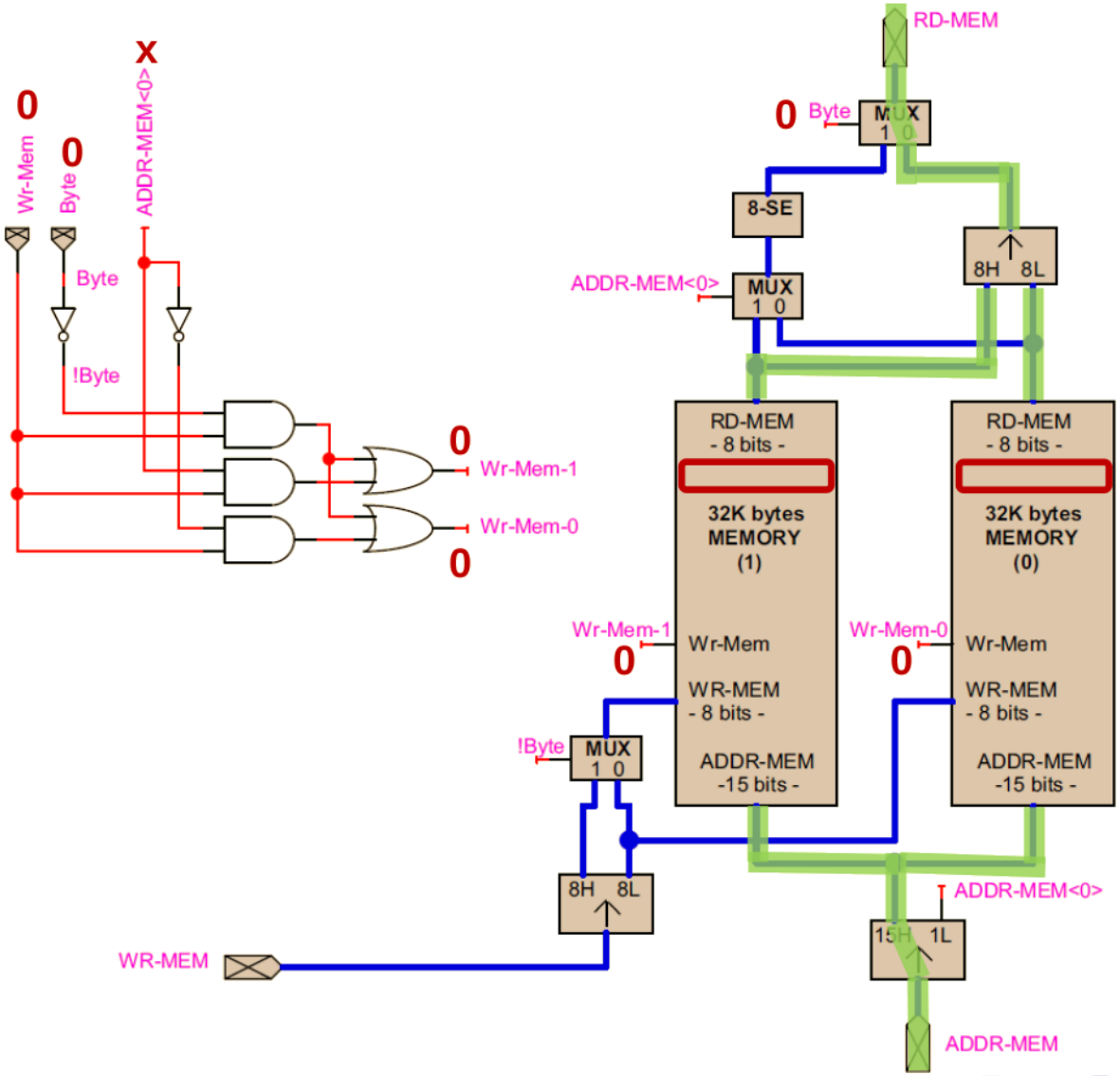


Implementació completa *byte* o *word*



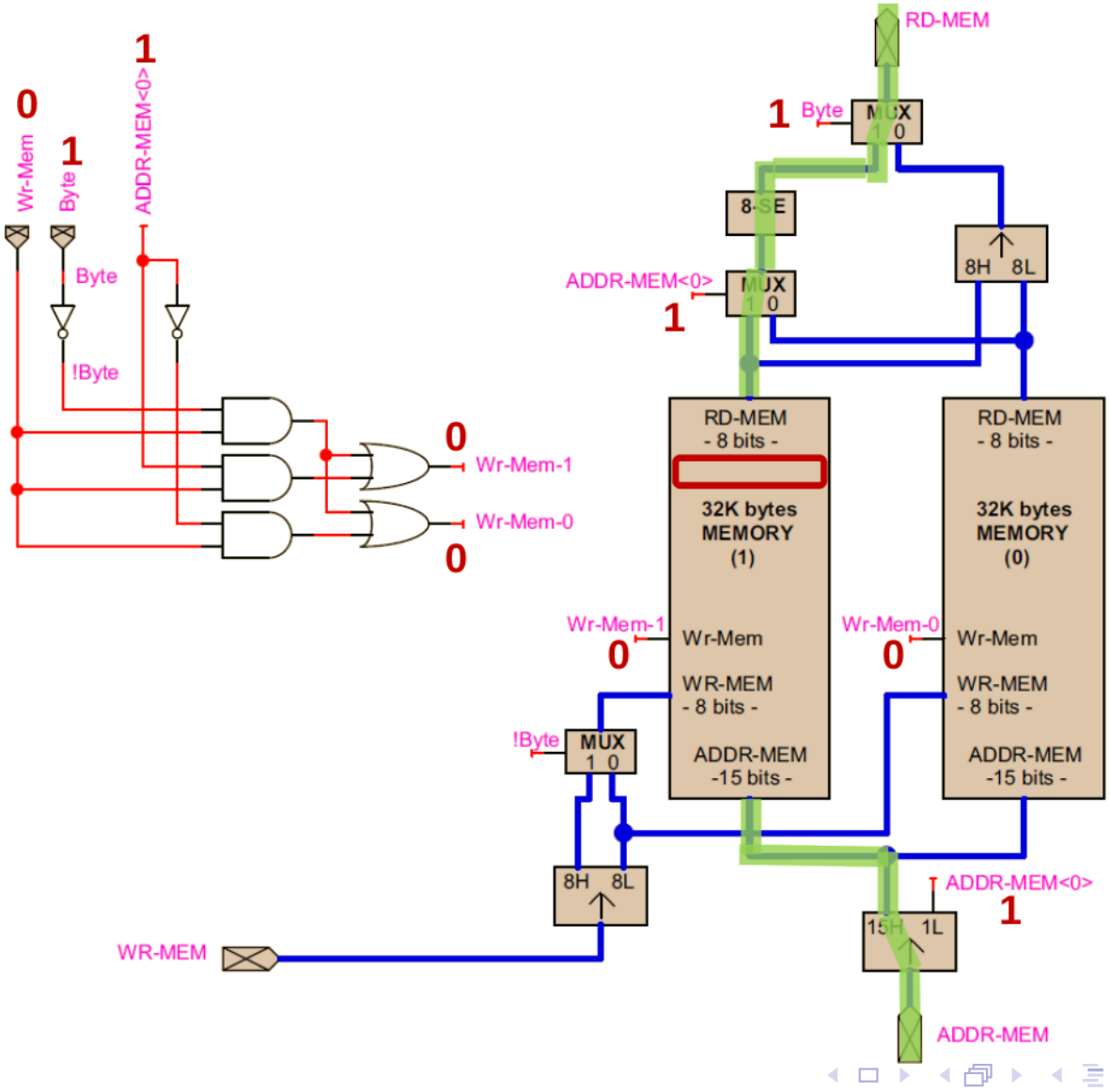
Camí LD Rd, N6(RA)

$$Rd \leftarrow MEM_w[((Ra+SE(N6))\&(\sim 1)]$$



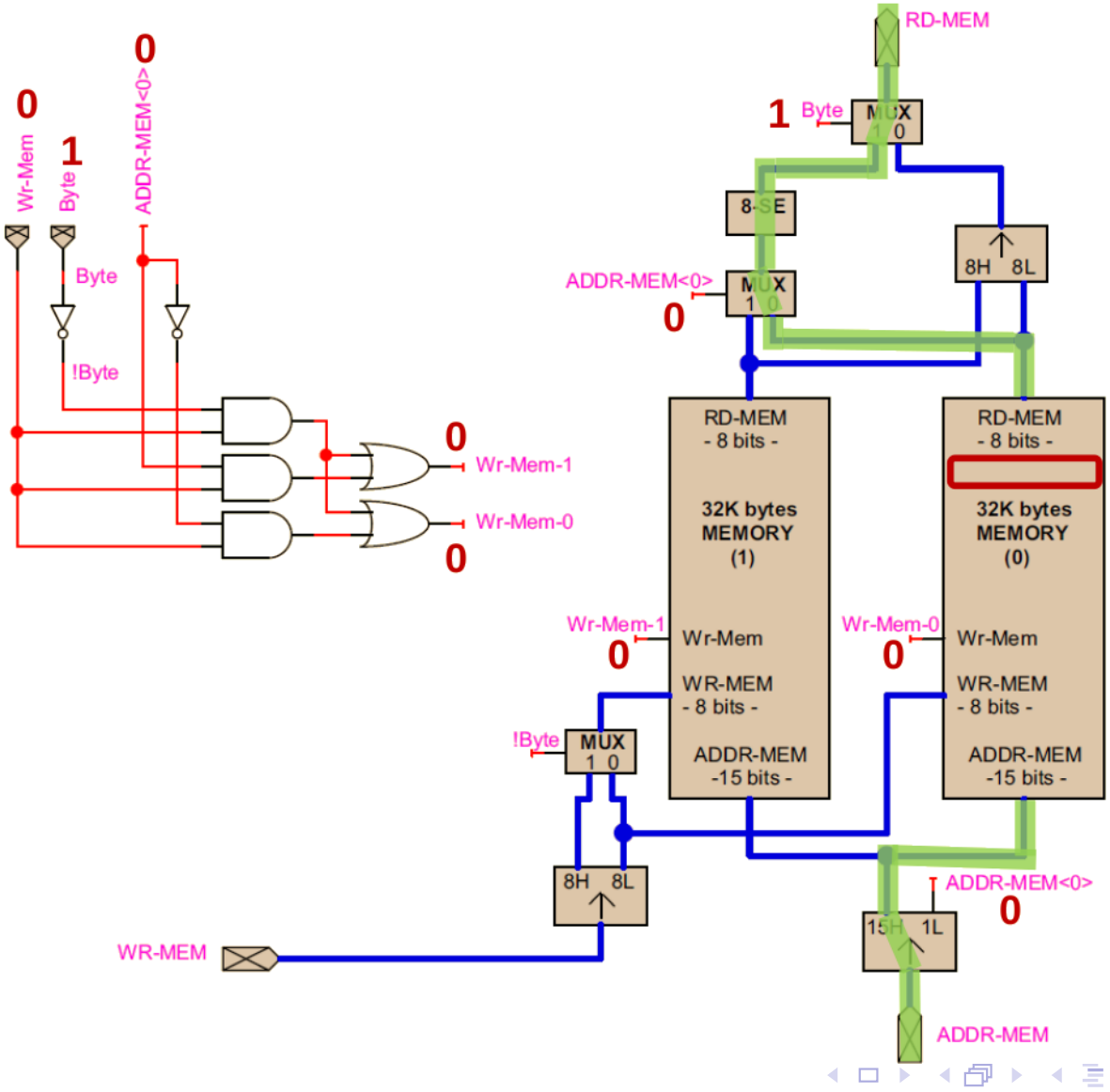
Camí LDB Rd, N6(RA) (adreça senar)

$$Rd \leftarrow SE(MEM_b[Ra+SE(N6)])$$



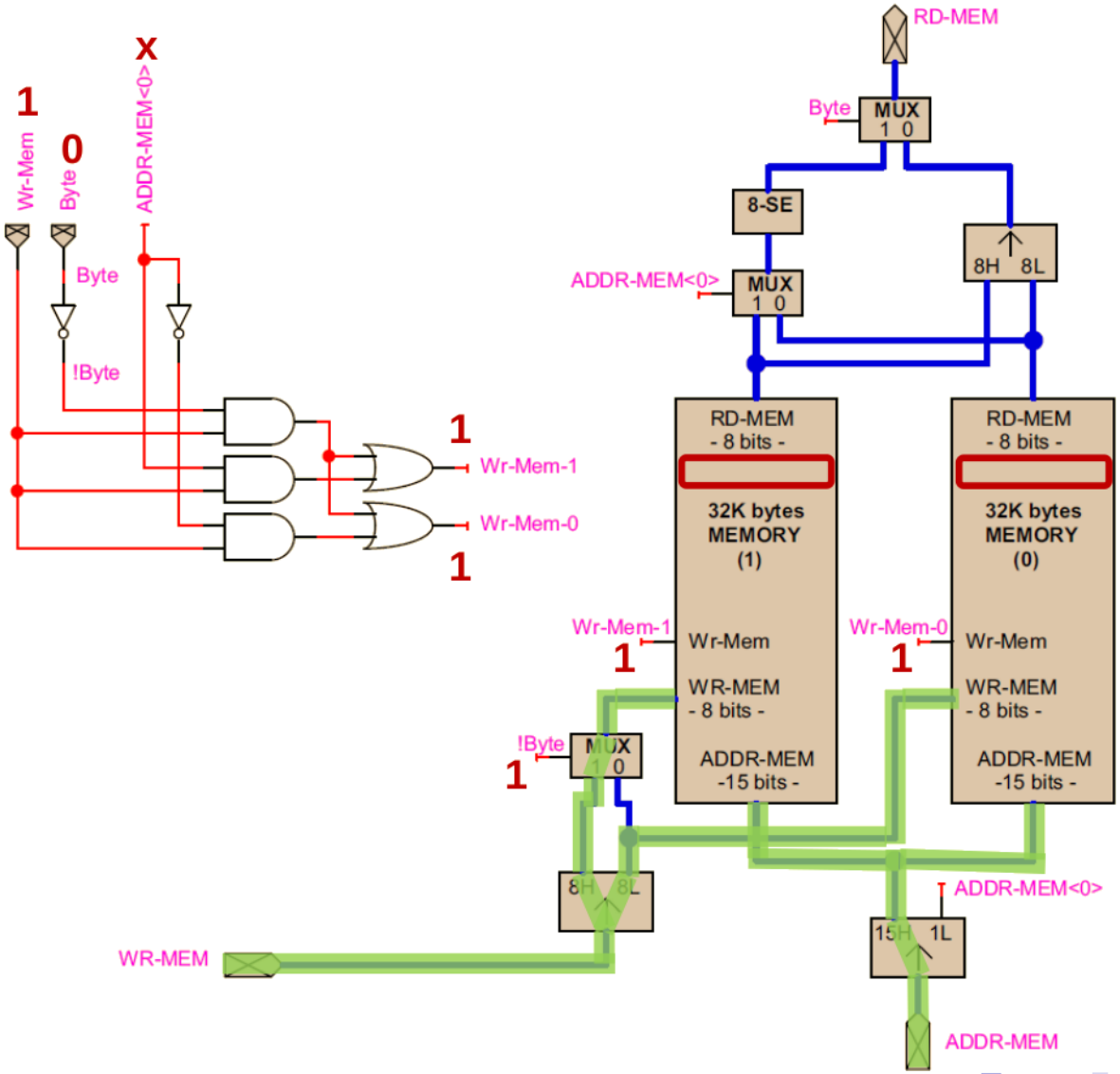
Camí LDB Rd, N6(RA) (adreça parell)

$$Rd \leftarrow SE(MEM_b[Ra+SE(N6)])$$



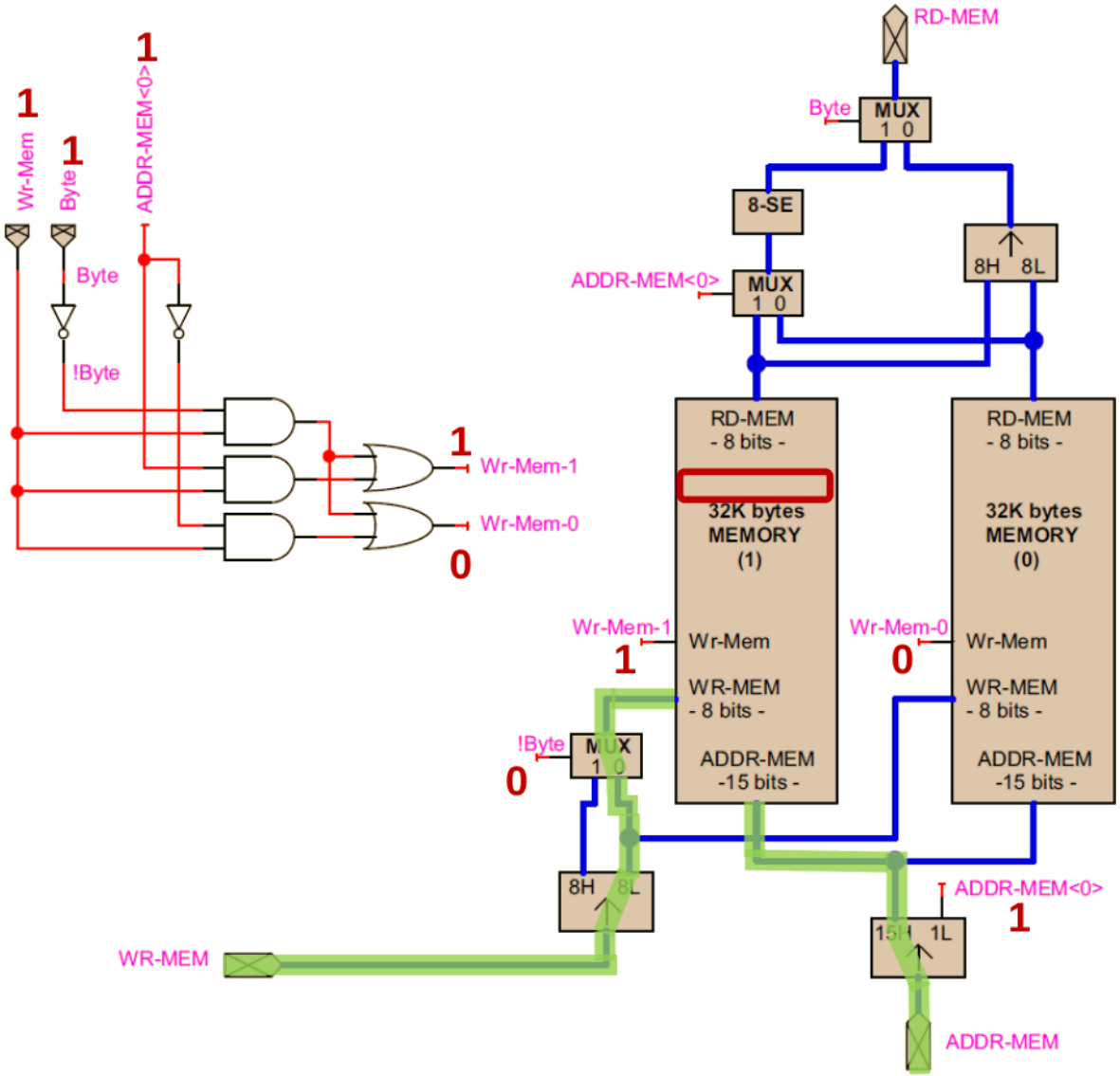
Camí ST N6(RA), Rb

$MEM_w[((Ra+SE(N6))\&(\sim 1)] \leftarrow Rb$



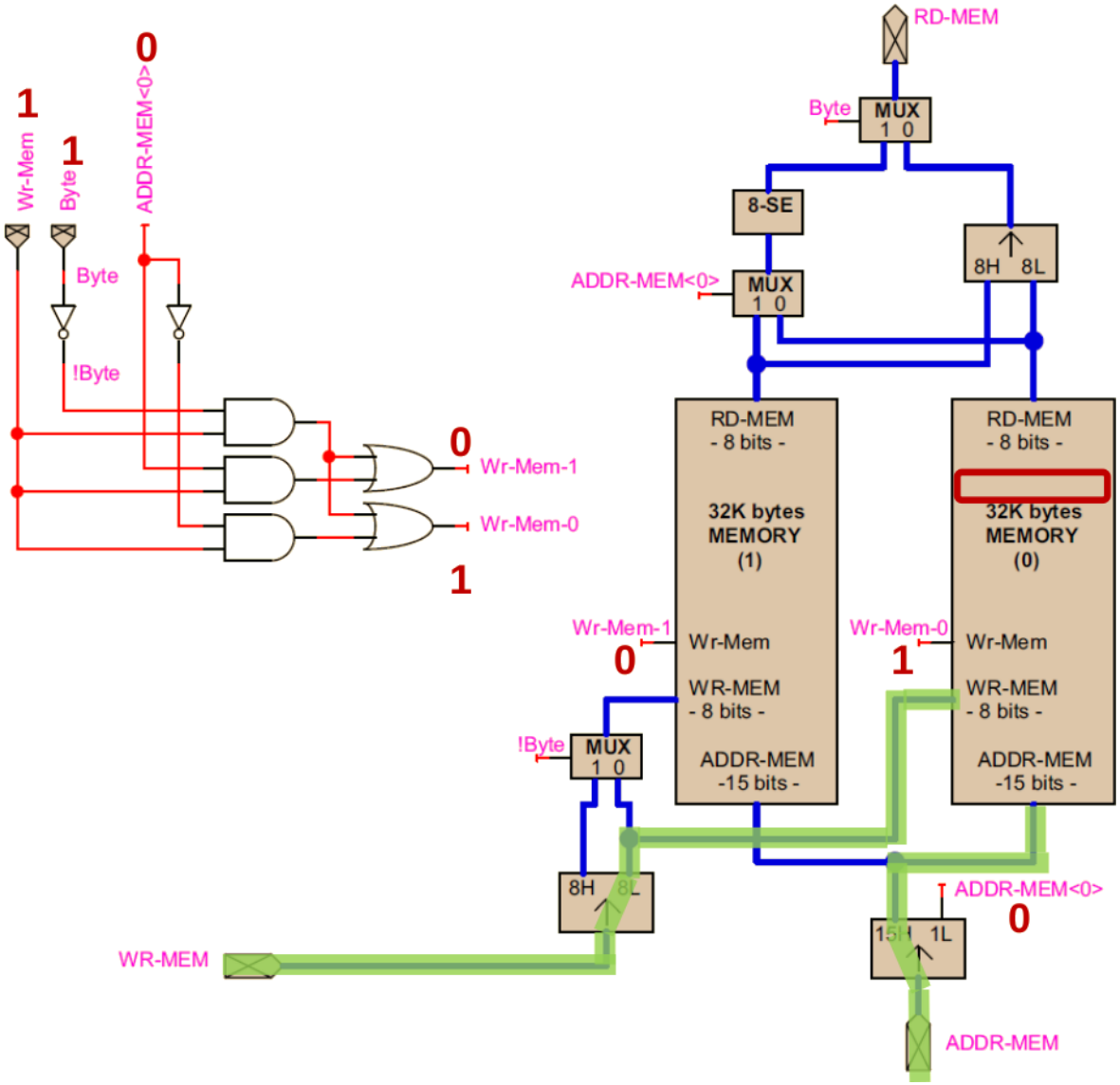
Camí STB N6(RA), Rb (adreça senar)

$MEM_b[Ra+SE(N6)] \leftarrow Rb<7..0>$



Camí STB N6(RA), Rb (adreça parell)

$MEM_b[Ra+SE(N6)] \leftarrow Rb<7..0>$



Índex

Exercicis: ensamblar/desensamblar

- `LDB R1, -3(R2)` $\rightsquigarrow$ `0101 010 001 111101` $\rightsquigarrow$ `0x547D`
- `ST 5(R6), R7` $\rightsquigarrow$ `0100 110 111 000101` $\rightsquigarrow$ `0x4DC5`
- `LD R5, -1(R6)` $\rightsquigarrow$ `0011 110 101 111111` $\rightsquigarrow$ `0x3D7F`
- `STB -2(R6), R1` $\rightsquigarrow$ `0110 110 001 111110` $\rightsquigarrow$ `0x6C7E`

Exercicis: modificació de l'estat

- Assumim que el contingut inicial de la memòria és tal que els *bytes* amb adreça parell contenen 0xFA i els d'adreça senar contenen 0x34.
- Assumim que $R0 = 0x0000, R1 = 0x0001, ...R7 = 0x0007$ i $PC = 0x4520$
- Quines modificacions a l'estat del computador provoquen les següents instruccions SISA (són independents, totes actuen sobre l'estat inicial)?
  - `LD  R3, 5(R2)`
  - `LDB R2, -6(R1)`
  - `LDB R2, -5(R1)`
  - `ST  2(R5), R4`
  - `STB 2(R1), R7`

Exercicis: modificació de l'estat

- Assumim que el contingut inicial de la memòria és tal que els *bytes* amb adreça parell contenen 0xFA i els d'adreça senar contenen 0x34.
- Assumim que $R0 = 0x0000, R1 = 0x0001, ...R7 = 0x0007$ i $PC = 0x4520$
- Quines modificacions a l'estat del computador provoquen les següents instruccions SISA (són independents, totes actuen sobre l'estat inicial)?
  - `LD R3, 5(R2)` $\rightsquigarrow$ `R3 ← 0x34FA; PC ← 0x4522;`
  - `LDB R2, −6(R1)` $\rightsquigarrow$ `R2 ← 0x0034; PC ← 0x4522;`
  - `LDB R2, −5(R1)` $\rightsquigarrow$ `R2 ← 0xFFFA; PC ← 0x4522;`
  - `ST 2(R5), R4` $\rightsquigarrow$ $Mem_w[0x0006]$ `← 0x0004; PC ← 0x4522;`
  - `STB 2(R1), R7` $\rightsquigarrow$ $Mem_b[0x0003]$ `← 0x07; PC ← 0x4522;`

Exercici: SISA $\Longleftrightarrow$ paraula control

- Donada una instrucció SISA, determinar la seva paraula de control
  - `LD R2, 10(R6)`

@A	@B	Rb/N	OP	F	-/i/l/a	@D	WrD	Wr-Out	Rd-In	Wr-Mem	Byte	TknBr	N (hexa)	ADDR-IO (hexa)

- Donada una paraula de control, determinar la instrucció SISA

@A	@B	Rb/N	OP	F	-/i/l/a	@D	WrD	Wr-Out	Rd-In	Wr-Mem	Byte	TknBr	N (hexa)	ADDR-IO (hexa)
1 0 0	1 1 1	0	0 0 0	1 0 0	x x	x x x	0	0	0	1	1	0	F F F D	X X

Exercici: SISA $\Longleftrightarrow$ paraula control

- Donada una instrucció SISA, determinar la seva paraula de control
  - `LD R2, 10(R6)`

@A			@B			Rb/N	OP		F			-/i/l/a		@D			WrD	Wr-Out	Rd-In	Wr-Mem	Byte	TknBr	N (hexa)				ADDR-IO	(hexa)
1	1	0	x	x	x	0	0	0	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	A	X	X

- Donada una paraula de control, determinar la instrucció SISA

@A			@B			Rb/N	OP		F			-/i/l/a		@D			WrD	Wr-Out	Rd-In	Wr-Mem	Byte	TknBr	N (hexa)				ADDR-IO	(hexa)
1	0	0	1	1	1	0	0	0	1	0	0	x	x	x	x	x	0	0	0	1	1	0	F	F	F	D	X	X

  - `STB -3(R4), R7`

Exercici: omplir la ROM_CTRL_LOGIC

- Per files o per columnes o per interseccions

Dirección					Contenido																			
I<15>	I<14>	I<13>	I<12>	I<8>	Bnz	Bz	Wr-Mem	RdIn	WrOut	WrD	Byte	Rb/N	-fi/I/a1	-fi/I/a0	OP1	OP0	MxN1	MxN0	MxF	F2	F1	F0	Mx@D1	Mx@D0	
0	0	1	0	x																					ADDI
0	1	0	1	x																					LDB
1	0	0	1	1																					MOVHI

Exercici: omplir la ROM_CTRL_LOGIC

- Per files o per columnes o per interseccions

Dirección					Contenido																		
I<15>	I<14>	I<13>	I<12>	I<8>	Bnz	Bz	Wr-Mem	RdIn	WrOut	WrD	Byte	Rb/N	-/i/I/a1	-/i/I/a0	OP1	OP0	MxN1	MxN0	MxF	F2	F1	F0	Mx@D1	Mx@D0	
0	0	1	0	x	0	0	0	0	0	1	x	0	0	0	0	0	0	0	1	1	0	0	0	1	ADDI
0	1	0	1	x	0	0	0	0	0	1	1	0	0	1	0	0	0	0	1	1	0	0	0	1	LDB
1	0	0	1	1	0	0	0	0	0	1	x	0	0	0	1	0	0	1	1	0	1	0	1	0	MOVHI

Exercicis a entregar a Atenea

- Enunciat disponible a Atenea
  - `https://atenea.upc.edu/pluginfile.php/3603448/mod_assign/introattachment/0/Tema%2011%20-%20Exercicis%20en%20paper.pdf?forcedownload=1`
- Entrega a Atenea fins el dilluns 30/11
  - Format PDF

Índex

Conclusions

- Hem afegit un mòdul de memòria al computador
  - Hem ampliat amb 4 noves instruccions el LM SISA
- Hem completat el nostre computador Harvard unicicle
  - Al tema 12 estudiarem el seu temps de cicle
- Xuletari temes 8 al 11
  - `https://atenea.upc.edu/mod/resource/view.php?id=1673034`
- No oblideu realitzar el qüestionari ET11 i els exercicis (slide 61)

Referències I

Llevat que s'indiqui el contrari, les figures, esquemes, cronogrames i altre material gràfic o bé han estat extrets de la documentació de l'assignatura elaborada per Juanjo Navarro i Toni Juan, o corresponen a enunciats de problemes i exàmens de l'assignatura, o bé són d'elaboració pròpia.

[1] [Online]. Available: http://blog.modernmechanix.com/two-bytes-are-better-than-one-2/.

Introducció als Computadors

Tema 11: Memòria

http://personals.ac.upc.edu/enricm/Docencia/IC/IC11.pdf

Enric Morancho
(enricm@ac.upc.edu)

Departament d'Arquitectura de Computadors
Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

2020-21, 1^{er} quad.