Es. n.1 Pipeline non bloccante con più unità funzionali multiciclo

A: FADD (in 2T) M: FMUL (in 3T) D: FDIV (in 3T)

Sequenza

FADD **F20**,F2,F1
FDIV **F2**,F2,**F20**
FMUL **F5**,F1,**F2**
FMUL F5,**F5**,F3
FMUL **F1**,F9,F10
FADD F9,F3,**F1**

Stimare il numero di colpi di clock al di sotto del quale non è possibile scendere nell'esecuzione del codice assegnato, qualunque numero di RS,CRB e stadi di fetch e decode disponibili, e si motivi la risposta

essendoci presenti delle dipendenze di dato la sequenza più lunga:

FADD **F20**,F2,F1
FDIV **F2**,F2,**F20**
FMUL **F5**,F1,**F2**
FMUL F5,**F5**,F3

risultera per tanto

IF – ID – FADD – WB – FDIV – WB – FMUL – WB – FMUL- WB

devo considerare WB alla fine di ogni esecuzione in quanto solo alla fine di quella fase otterò il dato da usare nell'istruzione sucessiva. L'altra sequenza potrà essere eseguita in parallelo essendo indipendente dalla prima (risulterà cmq. più corta e quindi irrilevante)

OPERAZIONE	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17
FADD **F20**,F2,F1	IF	ID	$A0_E$	$A0_E$	WB												
FDIV **F2**,F2,**F20**		IF	ID	$D0_W$	$D0_W$	$D0_E$	$D0_E$	$D0_E$	WB								
FMUL **F5**,F1,**F2**			IF	ID	$M0_W$	$M0_W$	$M0_W$	$M0_W$	$M0_W$	$M0_E$	$M0_E$	$M0_E$	WB				
FMUL F5,**F5**,F3				IF	ID	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_E$	$M1_E$	$M1_E$	WB

Il minimo tempo di colpi di clock è 17 ossia 1 IF + 1 ID + 2Tck + 3Tck + 3Tck + 3Tck + 4 WB = 17Tck

Mostrare la dinamica dell'esecuzione nel caso di 1 CRB, 1 IF, 1 ID e 2 RS per ogni unità funzionale

OPERAZIONE	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23	T24
FADD **F20**,F2,F1	IF	ID	$A0_E$	$A0_E$	WB																			
FDIV **F2**,F2,**F20**		IF	ID	$D0_W$	$D0_W$	$D0_E$	$D0_E$	$D0_E$	WB															
FMUL **F5**,F1,**F2**			IF	ID	$M0_W$	$M0_W$	$M0_W$	$M0_W$	$M0_W$	$M0_E$	$M0_E$	$M0_E$	WB											
FMUL F5,**F5**,F3				IF	ID	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_E$	$M1_E$	$M1_E$	WB							
FMUL **F1**,F9,F10					IF	ID_S	ID_S	ID_S	ID_S	ID_S	ID_S	ID_S	ID_S	ID	$M0_R$	$M0_R$	$M0_E$	$M0_E$	$M0_E$	WB				
FADD F9,F3,**F1**														IF	ID	$A0_W$	$A0_W$	$A0_W$	$A0_W$	$A0_W$	$A0_E$	$A0_E$	WB	

Da notare :

- in T5, T6 c'è una dipendenza di dato, posso partire ad eseguire il clock dopo la fase di WB in quanto devo aspettare che il dato venga cambiato nella RS delle unità funzionali
- T6 ho le unità funzionali piene della MUL devo quindi stallare finche non arrivo a T13,T14; ricordo che lo spazione nelle RS è disponibile solo alla fine della WB.
- T15 sono in stato ready in quanto aspetto solo la liberazione dell'unità funzionale e non aspetto dati
- T17 non essendoci dipendenza di dato tra le due istruzioni nella fase di WB della prima posso già partire ad eseguire l'altra istruzione che è in READY

$$Speed\,Up = \frac{tck_{sequenziale}}{tck_{parallelo}}$$ nel nostro caso corrente SpeedUp = 20 / 23 = 0,86%

Nel caso sequenziale (no pipeline tomasulo) ho 1 IF + 1 ID + 2 * 2 ADD + 1 * 3 DIV + 3 * 3 MUL + 1 MEM + 1 WB = 20 Tck

Si supponga ora di poter apportare una delle seguenti modifiche all'architettura :

- **aumento di RS**
- **raddoppio del CRB per poter eseguire due WB per clock**

Quale è la migliore modifica che conviene apportare se si desidera ottenere il massimo speed up? Perchè ?

Dato che non ci sono stati di WB_W allora sicuramente posso già dire tra le due è meglio la prima che mi impedirà gli stalli da T6 a T13.

Per verifica possiamo disegnare la tabella

3 RS **Faccio stallare in WB le unità che hanno tempo di Tck minore**

OPERAZIONE	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23	T24
FADD **F20**,F2,F1	IF	ID	$A0_E$	$A0_E$	WB																			
FDIV **F2**,F2,**F20**		IF	ID	$D0_W$	$D0_W$	$D0_E$	$D0_E$	$D0_E$	WB															
FMUL **F5**,F1,**F2**			IF	ID	$M0_W$	$M0_W$	$M0_W$	$M0_W$	$M0_W$	$M0_E$	$M0_E$	$M0_E$	WB											
FMUL F5,**F5**,F3				IF	ID	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_E$	$M1_E$	$M1_E$	WB							
FMUL **F1**,F9,F10					IF	ID	$M2_E$	$M2_E$	$M2_E$	WB														
FADD F9,F3,**F1**						IF	ID	$A0_W$	$A0_W$	$A0_W$	$A0_E$	$A0_E$	WB_W	WB										

2 CRB

OPERAZIONE	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23	T24
FADD **F20**,F2,F1	IF	ID	$A0_E$	$A0_E$	WB																			
FDIV **F2**,F2,**F20**		IF	ID	$D0_W$	$D0_W$	$D0_E$	$D0_E$	$D0_E$	WB															
FMUL **F5**,F1,**F2**			IF	ID	$M0_W$	$M0_W$	$M0_W$	$M0_W$	$M0_W$	$M0_E$	$M0_E$	$M0_E$	WB											
FMUL F5,**F5**,F3				IF	ID	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_E$	$M1_E$	$M1_E$	WB							
FMUL **F1**,F9,F10					IF	ID_S	ID_S	ID_S	ID_S	ID_S	ID_S	ID_S	ID_S	ID	$M0_R$	$M0_R$	$M0_E$	$M0_E$	$M0_E$	WB				
FADD F9,F3,**F1**														IF	ID	$A0_W$	$A0_W$	$A0_W$	$A0_W$	$A0_W$	$A0_E$	$A0_E$	WB	

Lo speedup quindi sarà

- con 3RS 20 / 17 = 1,17 %
- con 2 CRB 20 / 23 = 0,86 %

Si supponga invece sia di poter apportare alla struttura base una delle modifiche indicate al punto 3 sia di poter aggiungere un'altra unità funzionale con 2 RS. Quale unità funzionale converrebbeaggiungere? Nella miglior soluzione trovata si mostri quanti clock si risparmierebbero rispetto al punto 2. Si indichi anche il CPI_{medio} che si otterrebbe.

Sempre basandosi sul fatto che ci sono delle dipendenze di dato imprescindibili l'unica modifica convincente è quella di aggiungere una MUL in modo da poter eseguire l'ultimo blocco di istruzioni in maniera parallela e raddoppiando il CRB in modo da eliminare lo stallo di WB in T13

Si risparmierebbero 23-17 = 6 clock.

$$CPI_{MEDIO} = \frac{N_{CLK}}{N_{ISTRUZIONI}}$$ Nel nostro caso CPI_2 = 23/6 = 3,8 e CPI_3 = 17 / 6 = 2,8

tengo in considerazione due modidiversi per inserire le istruzioni nelle RS delle unità funzionali

2RS , 2 CRB e 2 MUL **Quando inserisco le istruzioni nelle RS prima di passare alla seconda N : MUL (3T) riempio le RS della prima**

OPERAZIONE	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23	T24
FADD **F20**,F2,F1	IF	ID	$A0_E$	$A0_E$	WB																			
FDIV **F2**,F2,**F20**		IF	ID	$D0_W$	$D0_W$	$D0_E$	$D0_E$	$D0_E$	WB															
FMUL **F5**,F1,**F2**			IF	ID	$M0_W$	$M0_W$	$M0_W$	$M0_W$	$M0_W$	$M0_E$	$M0_E$	$M0_E$	WB											
FMUL F5,**F5**,F3				IF	ID	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_W$	$M1_E$	$M1_E$	$M1_E$	WB							
FMUL **F1**,F9,F10					IF	ID	$N0_E$	$N0_E$	$N0_E$	WB														
FADD F9,F3,**F1**						IF	ID	$A0_W$	$A0_W$	$A0_W$	$A0_E$	$A0_E$	WB											

2RS , 2 CRB e 2 MUL **Quando inserisco le istruzioni nelle RS le riempio bilanciando le due MUL**

OPERAZIONE	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23	T24
FADD **F20**,F2,F1	IF	ID	$A0_E$	$A0_E$	WB																			
FDIV **F2**,F2,**F20**		IF	ID	$D0_W$	$D0_W$	$D0_E$	$D0_E$	$D0_E$	WB															
FMUL **F5**,F1,**F2**			IF	ID	$M0_W$	$M0_W$	$M0_W$	$M0_W$	$M0_W$	$M0_E$	$M0_E$	$M0_E$	WB											
FMUL F5,**F5**,F3				IF	ID	$N0_W$	$N0_W$	$N0_W$	$N0_W$	$N0_W$	$N0_W$	$N0_W$	$N0_W$	$N0_E$	$N0_E$	$N0_E$	WB							
FMUL **F1**,F9,F10					IF	ID	$M1_E$	$M1_E$	$M1_E$	WB														
FADD F9,F3,**F1**						IF	ID	$A0_W$	$A0_W$	$A0_W$	$A0_E$	$A0_E$	WB											

Es. n.2 Gerarchie delle memorie e DMAC

1.a Progettazione di P16

Si dica su quante posizioni dello spazio di I/O viene mappata P16.

Le possibili richieste alla periferica P16 (per semplicità si può mappare a 400H)

- Accesso LB A1 = 0 LB# = 0 UB# = 1 400H
- Accesso UB A1 = 0 LB# = 1 UB# = 0 401H
- Accesso (se il bus lo permette)16bit A1 = 0 LB# = 0 UB# = 0 -----
- Lettura dello Stato A1 = 1 LB# = 0 UB# = 1 402H

Un dispositivo accessibile attraverso il bus occupa "n" posizioni dello spazio di indirizzamento, dove "n" rappresenta il numero di oggetti di 8 bit indirizzabili all'interno del dispositivo. "n" inoltre dovrà essere una potenza di 2**k (con k piccolo)!

Data questa definizione è evidente che questo dispositivo P16 occupa 4 posizioni. [Volendo LB e UB si possono vedere come A0]

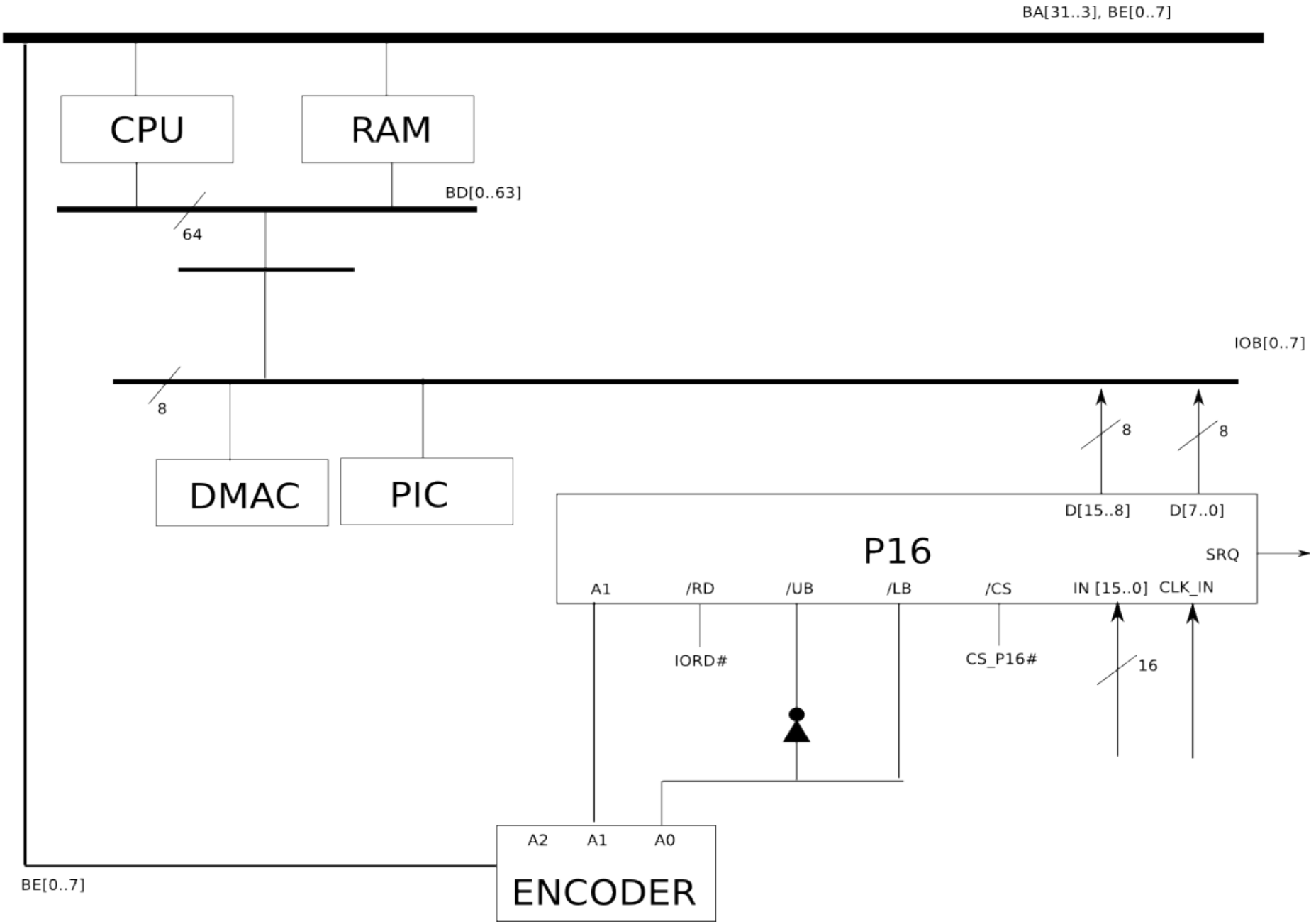
Si mostri come P16 può essere interfacciato al bus di I/O da 8 bit nel caso debba essere gestito solo dalla CPU e non dal DMAC.

Pensando come sopra che P16 si trovi a 400H

$$CS_{P16} = BA10 \cdot \overline{BA9}\,\overline{BA8}$$

Per l'encoder

BE	A2 A1 A0	
0	0 0 0	LB
1	0 0 1	UB
2	0 1 0	Stato
3	0 1 1	


BA[31..3], BE[0..7]
CPU
RAM
BD[0..63]
64
IOB[0..7]
8
DMAC
PIC
8
8
D[15..8]
D[7..0]
P16
SRQ
A1
/RD
/UB
/LB
/CS
IN [15..0]
CLK_IN
IORD#
CS_P16#
16
A2
A1
A0
ENCODER
BE[0..7]

Si scriva un frammento di codice assembler che legge il contenuto di P16 e lo porta in ax, e si mostrino le forme d'onda dei segnali del bus più significativi

Dato che non è presente un DMAC o un PIC allora posso accedere alla periferica solo a POLLING e non ad INTERUPT

```
polling:    IN AL, 402H          ; in 402H ho BE2# = 0
            CMP AL, 1
            JNZ polling          ; se lo stato è 1 allora posso procedere nella lettura
            IN AL, 400H          ; porta bassa
            IN AH, 401H          ; porta alta
```

.. disegno segnali significativi ....

1.b Gerarchie delle memorie

Si supponga di avere un sistema con cache abilitate e memoria virtuale (paging) disabilitato. Si ha un segmento D mappato all'indirizzo fisico 00020400H e contenente tre vettori A,B e V di 12 word (da due byte) ciascuno. Il sistema riceve A e B da P16 in DMA, calcola la somma dei due vettori A e B e mette il risultato in V.

Si indichi il contenuto del descrittore di D nell'ipotesi che D contenga solo i 3 vettori A, B e V

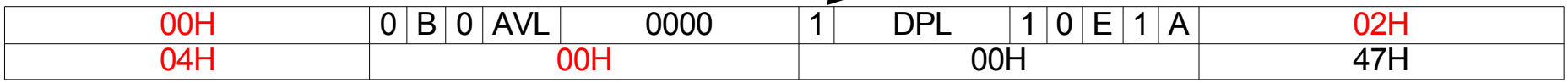
A, B e V sono ognuno di 12 * 2 = 24 Byte ossia D sarà grande 24 + 24 + 24 = 72 Byte

72 in esadecimale è 48H da cui si ricava che il limite del segmento sarà 48H -1 = 47H

LIMITE = 00 40H

BASE = 00 02 04 00H

Vale 1 perchè è presente in memoria fisica IF= 020400H

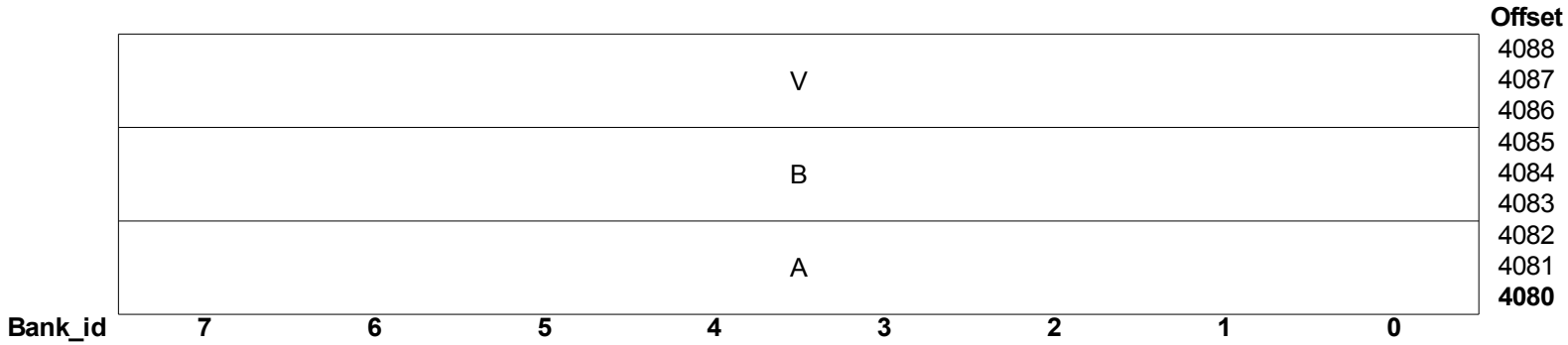


Supponendo che il Pentium disponga di una cache dati a 2 vie da 8KB complessivi e linee da 32byte, gestisca con stato MESI e con politica di scrittura WRITE AROUND in caso di miss. Supponendo di abilitare la cache:

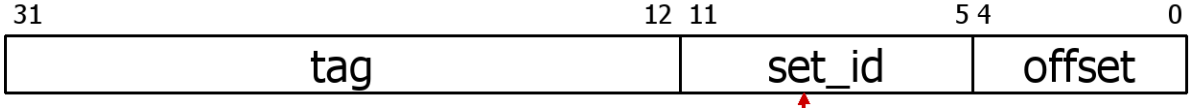
Si scrivano i valori di *tag* e *set_id* associati ai tre vettori A,B e V

I tre vettori in memoria fisica saranno cosi strutturati:

020400H => offset pari a 4080 (ossia 020400 - 0 / 8)



Il ***line_id*** (*tag* + *set_id*) sono i primi 27 bit , i rimanenti 5 sono l'offset



A	00020**400**H	00020**417**H	considerando ora i valori di line_id	020400H / 20H (ossia 32 = 2^5) = **001020**H x tutte e 12 le parole
B	00020**418**H	0002042**F**H	considerando ora i valori di line_id	**001020**H per 4 parole e **001021**H per 8 parole
V	00020**430**H	00020**447**H	considerando ora i valori di line_id	**001021**H per 8 parole e **001022**H e per 4 parole

Si avrà quindi per A :

00020400H 00020417H

rimuovo i primi 20

400H 417H

e tolgo i primi 5 bit (o divido per 20H direttamente)

0100 000~~0 0000~~ **0100 000**~~1 0111~~

ottengo

20H 20H

Stesso procedimento per B e C tenendo presente che sono spezzati sicuramente su due linee

B => *00020418*H => *418*H => **20**H
=> *0002042F*H => *42F*H => **21**H

V => *00020430*H => *430*H => **21**H
=> *0002044F*H => *44F*H => **22**H

Per riassumere

A set_id 20H

B set_id 20H x i primi 4 elementi del vettore
21H x i restanti 8 elementi

V set_id 21H x per i rpimi 8 elementi
set_id 22H x i restanti 4 elementi

Si indichino i contenuti significativi del TSS del task che calcola V e si scriva in assembler il codice che calcola V

E' evidente che le due cose sono in relazione

SOLUZIONE 1

```
; V[x] = A[x] + B[x]   con x da 11 a 0
; init di DS e SS              ; ho in DS l'indirizzo del segmento D
MOV  AX,30H                    ; V si trova a partire da 30H
MOV  BX,18H                    ; B si trova a partire da 18H
MOV  CX,0BH                          ; contatore di parole
ciclo: MOV  DX, [CX*2]         ; muovo in DX (16 bit) il contenuto di DS + CX * 2
```

```
        ADD   DX, BX[CX*2]      ; muovo in DX (16 bit) il contenuto di DS + BX + CX * 2
        MOV   AX[CX*2],DX       ; memorizzo DX che contiene la somma nella cella del vettore V
        DEC   CX                ; diminuisco il contatore
        CMP   CX, -1            ; verifico se sono a -1
        JNZ ciclo               ; ciclo finchè CX non è -1
```

I/O Map Addres	
	LDT Segment Selector
	GS
	FS
	DS
	SS
	CS
	ES
EDI	
ESI	
EBP	
ESP	
EBX	
EDX	
ECX	
...EAX..	AH / AL
EFLAGS	
EIP	
CR3 (PDBR)	
	SS2
ESP2	
	SS1
ESP1	
	SS0
ESP0	
	Previous Task Link

SOLUZIONE 2

```
        ; V[x] = A[x] + B[x]   con x da 0 a 11
        ; init di DS e SS
        MOV   AX,30H            ; V si trova a partire da 30H
        MOV   BX,18H            ; B si trova a partire da 18H
        MOV   CX,0              ; contatore di parole
ciclo:  MOV   DX, [CX*2]        ; muovo in DX (16 bit) il contenuto di DS + CX * 2 ossia A[CX]
        ADD   DX, BX[CX*2]      ; sommo a DX (ossia A[CX]) il contenuto di DS + BX + CX * 2 ossia B[CX]
        MOV   AX[CX*2],DX       ; memorizzo DX che contiene la somma nella cella del vettore V
        INC   CX                ; diminuisco il contatore
        CMP   CX, 0BH           ; verifico se sono a 11
        JNZ ciclo               ; ciclo finchè CX non è 11
```

il TSS ovviamente è uguale a meno dei valori iniziali di CX

Gli elementi significativi del primo caso saranno :

DS		*Selettore nella LDT di D*
	EDI	
	ESI	
EBX		0018H
EDX		DX
ECX		000BH
EAX		0030H

DX varia nel tempo e conterrà tuttti gli elementi della somma.

Si calcoli il numero di cicli di WB eseguiti dalla CPU durante i trasferimenti in DMA, e si indichi come è stato calcolatoil risultato ottenuto, nell'ipotesi che inizialmente le linee che contengono A,B e V siano invalide.

Se le linee nella cache di A,B e V sono Invalide la CPU non esegue cicli di WB in quanto nel protocollo MESI da I si rimane in I se un'altro master scrive riferendosi agli stessi dati.

Si indichi la sequenza di accessi alla memoria nel calcolo degli elementi di V, e si calcoli con precisione la MISS RATE. Si indichi inoltre lo stato MESI della cache dati immediatamente prima e dopo il calcolo di V. Si dica quanti cicli di burst e quanti cicli singoli vengono eseguiti dalla CPU durante il calcolo di V.

(riferendomi alla soluzione 1)

1. carico l'elemento A[11] in DX
2. sommo a DX , il contenuto di B[11]
3. metto DX in V[11]

nella fase 1 ho 1 miss in lettura e poi 11 hit , nella fase 2 ho 4 hit, 1 miss in lettura e poi altre 8 hit; nella terza fase dato che devo scrivere ho 4 miss in scrittura (write around) e 8 hit in scrittura con relativa WB in futuro.

Gli accessi saranno 12 + 12 letture + 12 scritture = 36

$$MISS_{RATE} = \frac{NUM_{MISS}}{NUM_{ACCESSI}}$$ Nel nostro caso Miss_rate = 6 / 36 = 0,16 ossia 16%

Lo stato delle linee di cache prima del calcolo di V

	VIA 0			VIA 1		
set_id	tag	stato	dati	tag	stato	dati
22H	00020	I	V[8..11]			
21H	00020	I	B[4..11] , V[0..7]			
20H	00020	I	A[0..11], B[0..3]]			

e dopo il calcolo di V

	VIA 0			VIA 1		
set_id	tag	stato	dati	tag	stato	dati
22H	00020	I	V[8..11]			
21H	00020	M	B[4..11] , V[0..7]			
20H	00020	E	A[0..11], B[0..3]]			

Durante il calcolo della CPU devo eseguire 2 miss in lettura quindi 2 cicli BURST generati dalla CPU , ho poi 4 miss in scrittura che generano 4 cicli singoli (write around) e per finire avrò in futuro quando dovrò sostituire la linea una fase di WB.

1.c progetto dell'hardware

Il sistema ora funziona con cache e memoria virtuale disabilitate. La ricezione di A e B da P16 è gestita da due transazioni controllate dal DMAC programmato con BCR = 0BH (un trasferimento elementare da 16 bit per ogni elemento di A o B). Si progetti il sistema con P16, DMAC interfacciati ad A ed in particolare :

Si disegni lo schema a blocchi del sistema e si indichino le parole di programmazione del DMAC e dei registri esterni, motivando le scelte fatte

L'indirizzo dovrebbe essere 00020400H quindi se uso 2 piedini invece di 3 allora lo devo dividere per 2 prima di inserirlo.

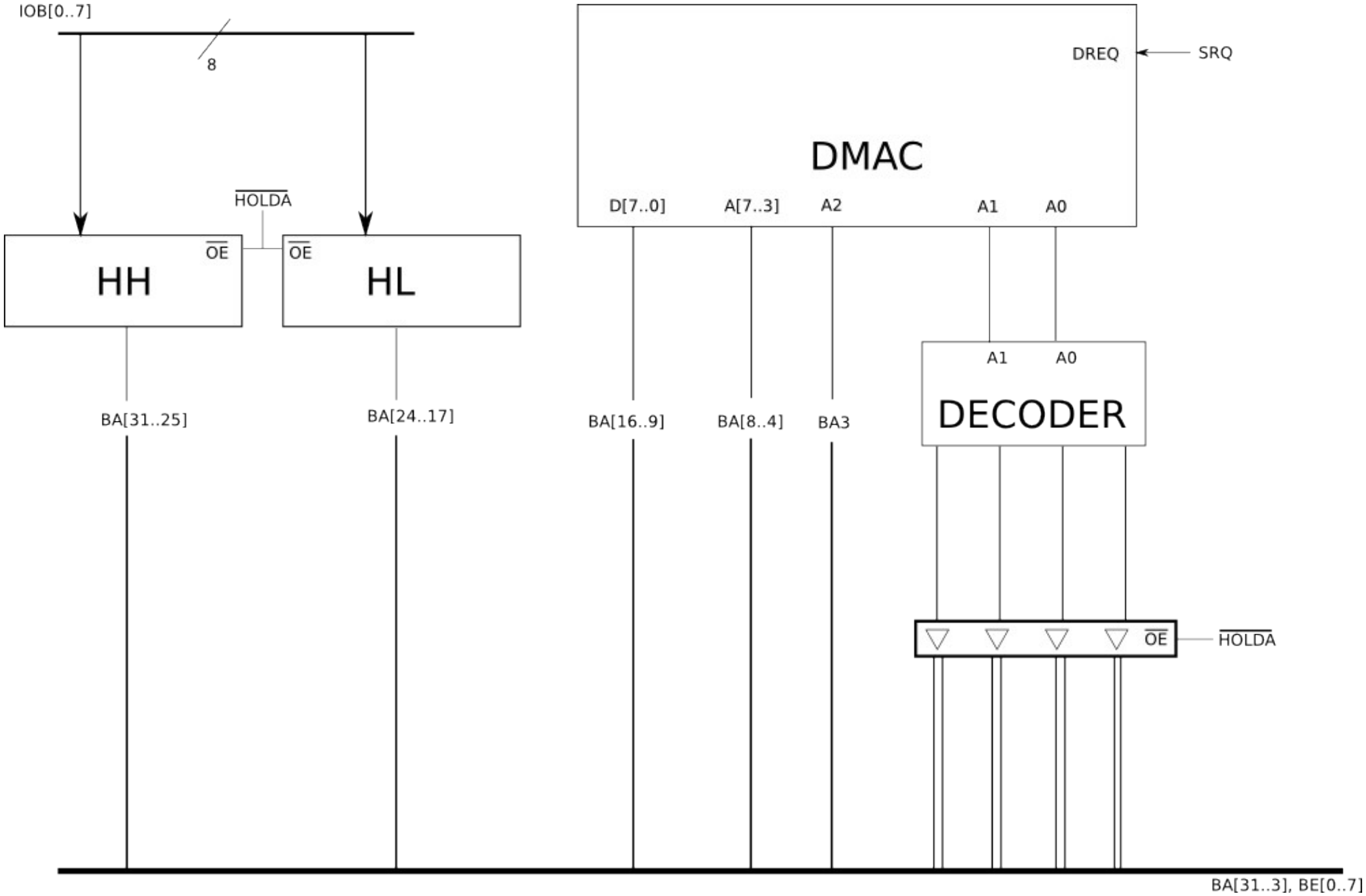
BAR = 0200H

HL = 01H

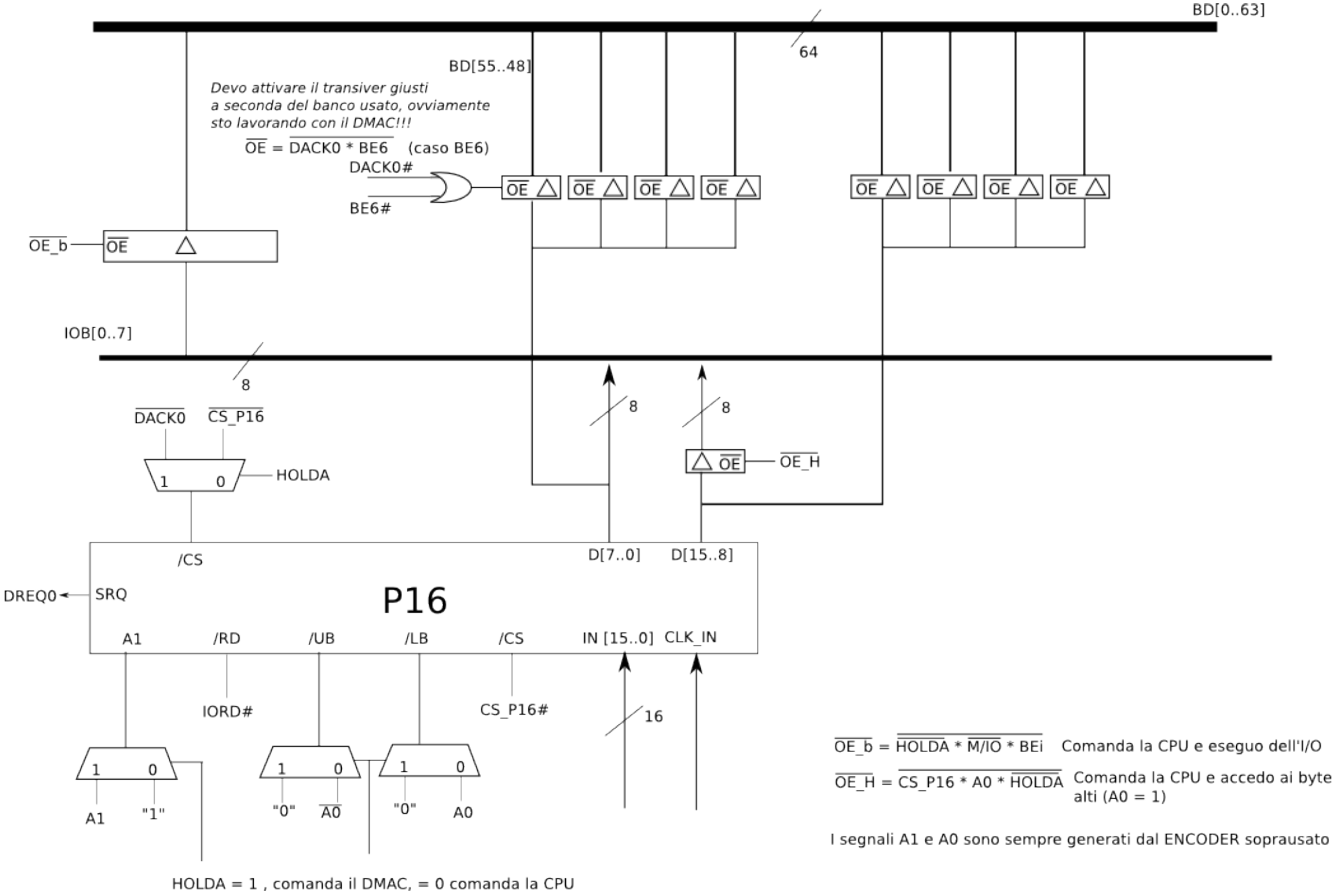
HH = 00H

Mode Register = single mode, autoincrement, autoinit disable, write, channel 0 = 044H

per capire meglio la cosa posso pensare alla necessità di incrementare l'offset di 1 ogni 4 invece che ogni 8 (default del dmac), quindi penso ad un contatore x 4, quale è il bit nel contatore per 4 che cambia ogni 4? è A2, allora dovrò sfruttare tale bit come primo bit dell'offset, ecco che lego A2 a BA3 , il resto viene di conseguenza!


IOB[0..7]
8
HOLDA
HH
OE
HL
OE
BA[31..25]
BA[24..17]
DMAC
DREQ
SRQ
D[7..0]
A[7..3]
A2
A1
A0
BA[16..9]
BA[8..4]
BA3
A1
A0
DECODER
OE
HOLDA
BA[31..3], BE[0..7]

Si mostri come un secondo bridge attraverso il quale P16 può essere interfacciato al bus di memoria BD[0..63] al fine di ottenere il trasferimento di 2 byte da P_16 alla memoria per ogni trasferimento elementare fly-by effettuato dal DMAC.



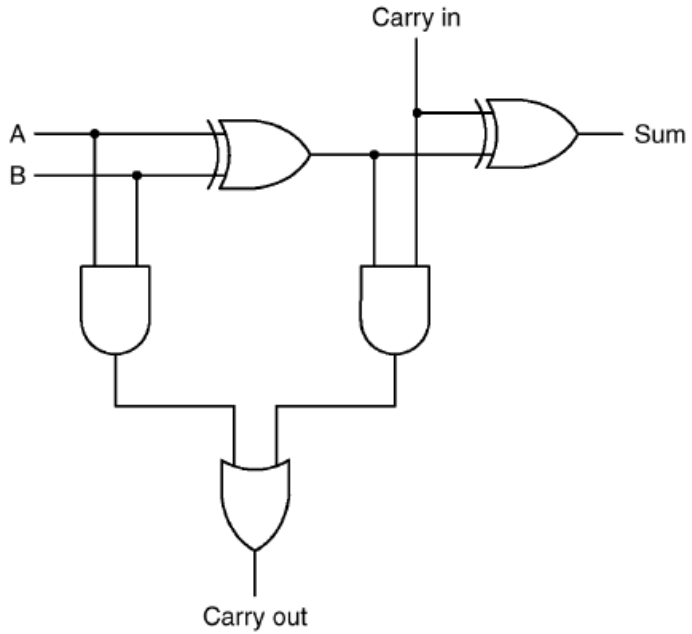
Si mostri come P_16 viene interconnesso al bus di I/O attraverso il quale se ne può leggere lo stato e al secondo bridge utilizzato solo in DMA, stando attenti al datapath e mostrando come devono essere pilotati i pin di P_16 (vedi sopra)

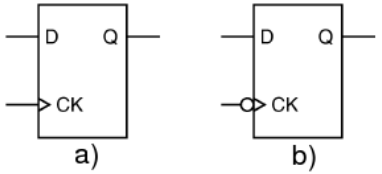
Si indichi come vanno generati da parte del DMAC i segnali del bus degli indirizzi (vedi sopra)

Si dica come vanno modificate le espressioni di controllo standard del bridge (vedi sopra)

Addizionatore Completo (Full Adder)

A	B	Carry in	Sum	Carry out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1





• I **Flip-Flop** commutano sui fronti del clock:

a) Commuta sul *fronte di salita*

b) Commuta sul *fronte di discesa*