ESERCIZI DI ARCHITETTURA – WALKTHROUGH

MICROISTRUZIONI - Metodi di indirizzamento:

1. %EAX: indirizzamento diretto a registro.
   Il valore viene prelevato dal registro
2. (%EAX), indirizzamento indiretto a registro.
   Il valore si trova nell'indirizzo di memoria puntato da EAX. Bisogna quindi mandare EAX nel MAR, aspettare la memoria (WMFC) e poi prendere il valore in MDR.
3. $*n*(%EAX), indirizzamento indiretto con spiazzamento.
   Devo sommare il valore *n* al valore puntato da EAX per ricavarmi l'indirizzo

MICROISTRUZIONI - Istruzioni e passi logici per l'esecuzione:

1. ADD src, dest
   Viene sommato il valore di src a dest e il valore ottenuto viene caricato in dest
   1. Preleva src e mettilo in V
   2. Metti dest sul BUS, seleziona ADD per la ALU e ottieni la somma in Z
   3. Prendi Z e mettilo in dest
2. SUB src, dest
   Viene sottratto il valore di src a dest e il valore ottenuto viene caricato in dest
   1. Uguale ad ADD, bisogna però fare il complemento a 2 di src
3. J value
   Indica un salto verso value che può essere un indirizzo, un'etichetta, un valore. Esistono diversi tipi di JUMP, descritti dalle lettere successive a J: Z (jump if zero), E (jump if equal), L (jump if less), G (jump if great). A queste è possibile aggiungere la N (davanti alla lettera identificativa del tipo di salto) per ottenere la negazione. Esiste anche il salto incondizionato, JMP
   1. Esegui il CMP, se non è soddisfatta la condizione di salto END
   2. Ottenuto il valore sommarlo a PC e metterlo su PC stesso
      NB: Se il salto è assoluto prendi il valore e mettilo in PC
4. PUSH value
   Impila sullo stack il valore value.
   1. Decrementa ESP di 4
   2. Inserire il nuovo valore in ESP e nel MAR
   3. Prendere value e inserirlo in MDR e dare il comando WRITE
   4. Dopo WMFC si ha l'END
5. POP dest
   Preleva il valore sulla cima dello stack e lo inserisce in dest
   1. Incrementa ESP di 4
   2. Inserire il nuovo valore in ESP e nel MAR
   3. Dare READ e attendere WMFC
   4. Prendere il valore nel MDR e metterlo in dest
6. CALL value
   La CALL indica una chiamata a funzione; è sempre preceduta dal PUSH di PC
   1. PUSH di PC
   2. Incrementa ESP di 4
   3. Inserire il nuovo valore in MAR
   4. Caricare PC in MDR e dare WRITE
      Questo è necessario per salvare in una cella vuota dello stack il valore di ritorno della funzione per poi riottenerla con l'istruzione finale della funzione, RET.

FASE DI FETCH (SEMPRE UGUALE)

1) PC_{OUT}, MAR_{IN}, READ, SELECT[4], ADD, Z_{IN}
2) WMFC, Z_{OUT}, PC_{IN}
3) MDR_{OUT}, IR_{IN}

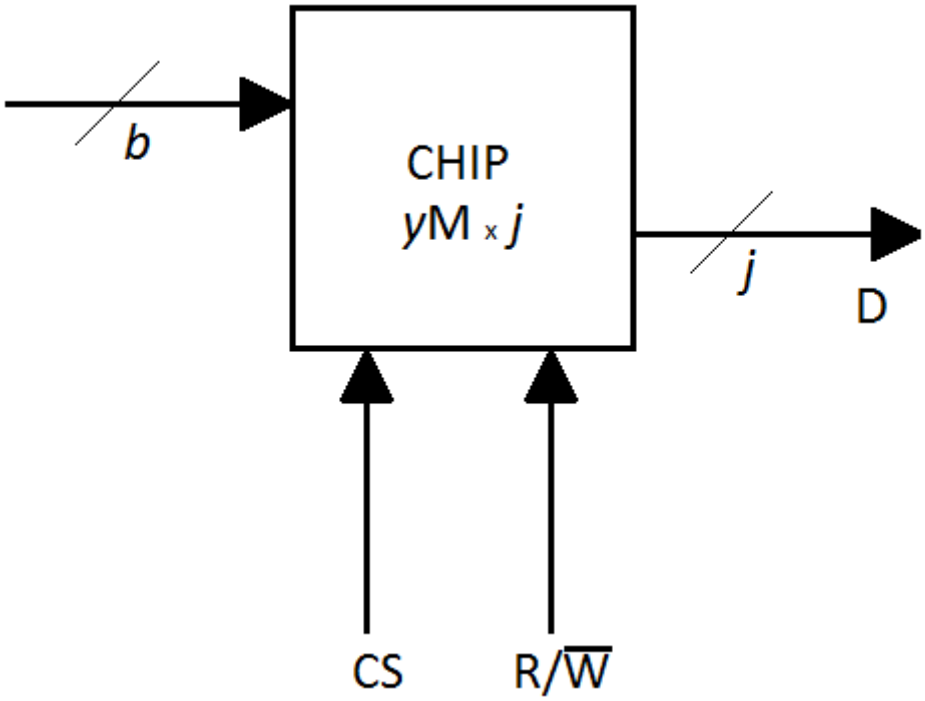
FASE INIZIALE DI CALL

4) ESP_{OUT}, SELECT[4], SUB, Z_{IN}
5) Z_{OUT}, ESP_{IN}, MAR_{IN}
6) PC_{OUT}, MDR_{IN}, WRITE
7) WMFC … (dipende se diretto o relativo)

DIMENSIONAMENTO DELLA RAM

Data una dimensione di RAM da creare (*g*), la dimensione a cui deve essere indirizzabile (*k*) e le caratteristiche del chip con cui crearla (*y*M x *j*)

1. Calcolare quanti bit servono per rappresentare la dimensione del chip *y*M
    a. Ai bit che servono per rappresentare M (20) sommo i bit necessari per rappresentare *y* e ottengo il numero *b*
2. Calcolare quanti bit servono per rappresentare la dimensione della RAM *g*
3. Calcolare quante colonne dovrà avere la matrice di chip
    a. Trasformo *j* da bit a byte (*j*/8)
    b. Eseguo *k*/*j* e ottengo quante colonne *c* dovrò avere
4. Calcolare quante righe dovrà avere la matrice di chip
    a. Eseguo *g*/*y* e ottengo quante righe *r* dovrò avere
5. Calcolare l'indirizzamento
    a. Calcolare quanti bit *z* servono per rappresentare *r*
    b. Sottrarre i bit appena ottenuti dal numero di bit trovati al punto 2 per verificare che i conti tornino. I bit rimanenti dovranno indirizzare i chip, quindi essere uguali a quelli trovati al punto 1.
6. Disegnare il tutto
    a. Disegnare il CHIP singolo



1 - disegno del chip

b. Disegnare lo schema degli indirizzi

bit per rappresentare *r*	*b* - indirizzamento CHIP

2 - schema indirizzamento

c. Disegnare lo schema completo della RAM
   1. Mettere un decoder che seleziona le righe e che ha come input *z* bit
   2. Aggiungere una linea input r/$\overline{w}$ che collega tutti i CHIP
   3. I CHIP sono collegati orizzontalmente da r/$\overline{w}$ e verticalmente dalla linea D

TROVARE LA DIMENSIONE DELL'INDIRIZZO DELLA RAM E DELLA CACHE

Abbiamo 3 campi: parola, blocco/pagina ed etichetta

1. Dimensione dell'indirizzo
   La dimensione dell'indirizzo si ottiene prendendo il numero di parole nella RAM e scomponendolo in potenza di 2. Avremo così 2^n con *n* che rappresenta la dimensione.
2. Dimensione del campo parola
   Bisogna considerare il numero di parole per blocco e trasformarlo in potenza di 2. Si ottiene così un numero 2^m con *m* che rappresenta la dimensione del campo parola.
3. Dimensione del campo blocco/pagina
   Bisogna dividere il numero di parole nella cache per il numero di parole per blocco per ottenere la dimensione della cache. Poi dividere la dimensione della cache per il tipo di set. Si ottiene così un numero *p* che rappresenta la dimensione del campo blocco/pagina.
4. Dimensione del campo etichetta
   Bisogna dividere il numero di parole nella RAM per il numero di blocchi. Si ottiene così un numero *q* che rappresenta la dimensione del campo etichetta.

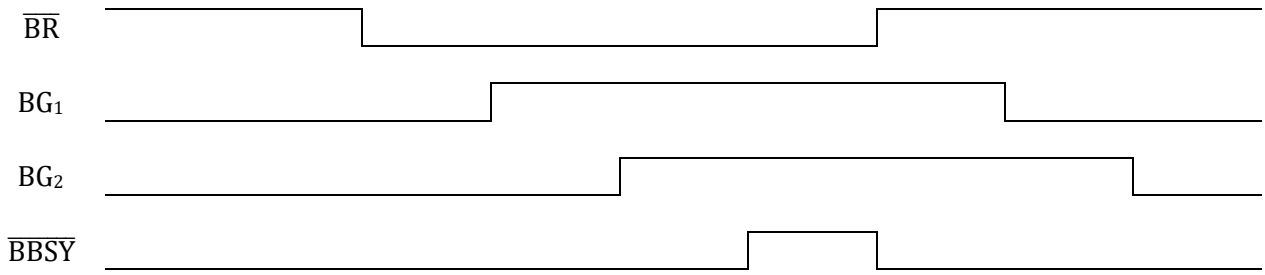
   $q + p + m = n$

   Per il disegno, farli nell'ordine q, p, m

$CPI_{medio} = 1$

1. Architettura a tre bus
2. Incremento del PC separato
3. Cache
4. Cache dati separata da cache del programma
5. $Cache_{HIT} \geq 95\%$
6. No operazioni sulla memoria
7. ALU che esegue le operazioni in un ciclo
8. Pipeline

Schema richiesta di utilizzo del BUS per dispositivo DMA_2



$T_{accesso}HDD = T_{seek} + T_{\omega} + T_{accessoBlocco}$

Località spaziale: dato un intervallo di memoria Δm, se al tempo ti accedo ad una parola in Δm, è altamente probabile che in ti + Δt acceda alla stessa zona di memoria.

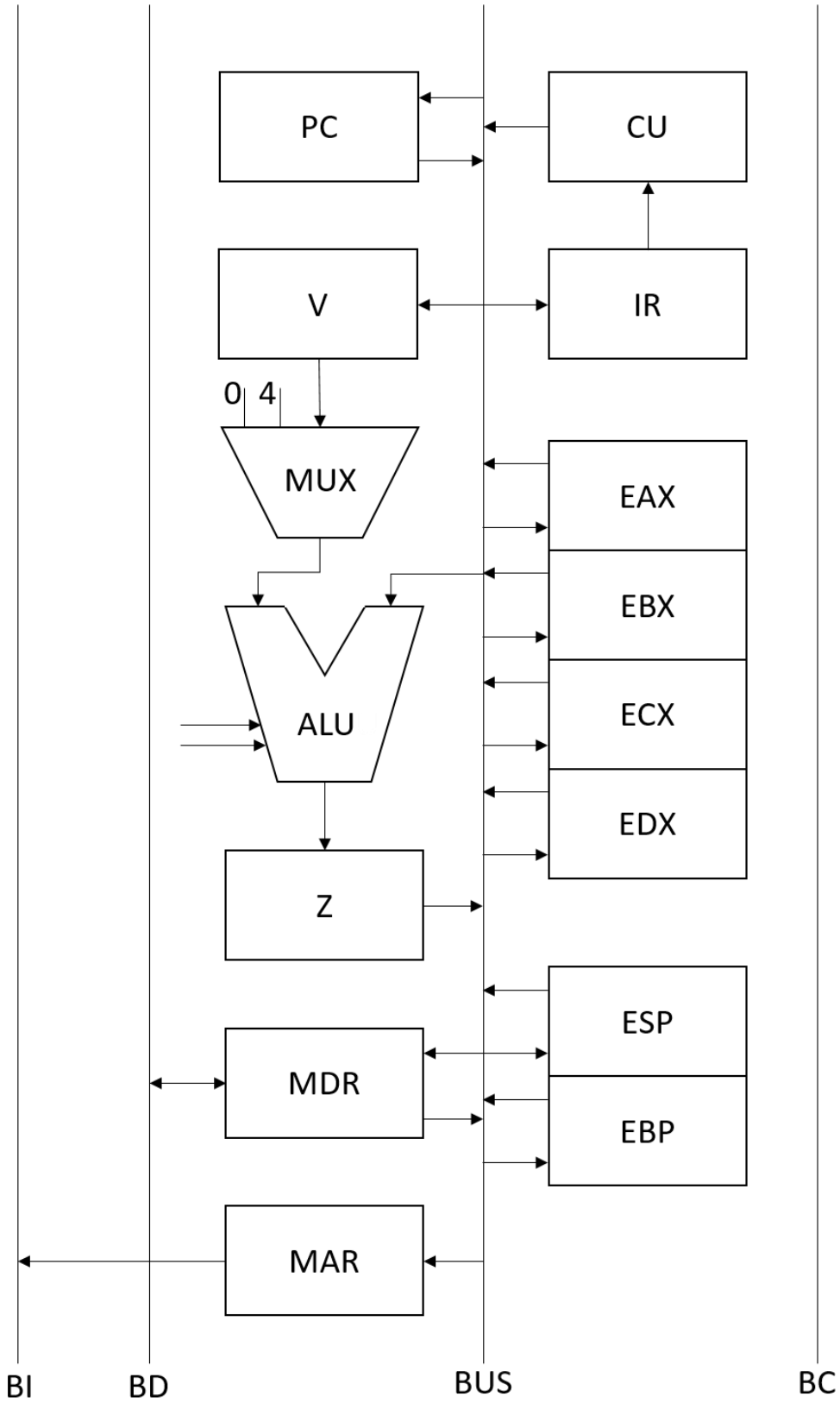
Località temporale: se in un intervallo di tempo Δt accedo ad una zona di memoria mi, è altamente probabile che acceda anche alla zona mi+Δm.

	$t_{accesso}$	Prob1	Prob2	Prob3
RAM	10T	99%	50%	5%
CACHE	T	1%	50%	95%
Principi		Niente	Località	Località + dati organizzati
		10T	5T	T

T_{exec} = N°istruzioni * CPI_{medio} * $1/F_{clock}$

Tassonomia di Flynn

	istruzioni		
dati		S	M
	S	SISD	MISD
	M	SIMD	MIMD


PC
CU
V
IR
0
4
MUX
EAX
EBX
ECX
EDX
ALU
Z
ESP
MDR
EBP
MAR
BI
BD
BUS
BC