Prof. Franco Fummi

ARCHITETTURA DEGLI ELABORATORI

SOLUZIONI degli APPELLI

dal 1999 al 2003

soluzioni di

Giorgia Jamila Michieli

Bogdan Maris

II PROVA PARZIALE 11/06/1999

Esercizio 1
Elencare le micro istruzioni relative alla completa esecuzione della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un solo BUS, che l'istruzione sia composta da una sola parola e che (%EBX) rappresenti un metodo di indirizzamento indiretto a registro (usare solamente le righe necessarie):
ADDL (%EBX), %EAX

1. PC_{OUT}, MAR_{IN}, READ, CLEAR Y, SET CARRY-IN, ADD, Z_{IN}
2. WMFC, Z_{OUT}, PC_{IN}
3. MDR_{OUT}, IR_{IN}
4. EBX_{OUT}, MAR_{IN}, READ
5. WMFC, EAX_{OUT}, Y_{IN}
6. MDR_{OUT}, ADD, Z_{IN}
7. Z_{OUT}, EAX_{IN}, END

Commenti:
1. fase di fetch, viene scaricato ed incrementato il PC
2. fase di fetch, viene messo in PC il valore PC+1
3. fase di fetch, escono i dati che vanno messi nel registro delle istruzioni
4. fase di decode, esce il valore contenuto in EBX e viene passato al registro MAR perchè possa far la codifica dell'indirizzo
5. si attende la risposta del MAR
6. il contenuto di EAX esce dall'MDR e viene sommato a EAX in quanto il metodo di indirizzamento è indiretto a registro
7. il risultato della somma dei due registri viene memorizzato in EAX. Fine dell'istruzione.

Esercizio 2 *Prima parte*
Si consideri una CPU dotata di memoria cache 4-associativa di 8K parole con 64 parole per blocco. Questa CPU è collegata ad una memoria RAM da 4M parole.
• Definire le dimensioni dell'indirizzo necessario a indirizzare tutta la memoria RAM e definire le dimensioni dei campi PAROLA, BLOCCO ed ETICHETTA in cui questo indirizzo può essere suddiviso. Motivare la risposta con un opportuno schema.

Cache	4-set associativa 8k parole
Blocco	64 parole
Ram	4M parole

Dimensioni dell'*indirizzo* necessario ad indirizzare tutta la memoria Ram:
dim. Ram 4M parole = 2^{22} quindi si hanno 22 bit per l'indirizzo

Dimensione del campo *parola*:
blocco = 64 parole = 2^6 quindi si hanno 6 bit per la parola

Dimensione del campo *set*:
dim. Cache / dim. Blocco = 8k/64 = $2^{13}/2^6 = 2^7$
Questo risultato va ora diviso per $2^{n^{\wedge}\ set}$, quindi: $2^7/2^4 = 2^3$ da cui si hanno 3 bit per il set

Dimensione del campo *etichetta*:
bit indirizzo della Ram - bit parola - bit set = Dimensione del campo etichetta
quindi
22 - 6 - 3 = 13

ETICHETTA	***SET***	***PAROLA***
13 bit	3 bit	6 bit

Esercizio 2 *Seconda parte*

• Si assuma che la cache appena descritta sia utilizzata per i dati, che sia inizialmente vuota e che utilizzi un algoritmo di sostituzione dei blocchi di tipo LRU (sostituzione dell'elemento meno utilizzato di recente). La CPU esegue un programma che accede in sequenza a tutti gli elementi di un array di 8320 parole (ogni elemento ha le dimensioni di una parola) che è memorizzato a partire dall'indirizzo 0. Questa operazione di scansione è effettuata all'interno di un ciclo che viene eseguito 5 volte.
Si assuma che il tempo di accesso alla cache sia di 1T e che il tempo di accesso alla memoria sia di 10T (entrambi i tempi si riferiscono alla lettura di una parola). Calcolare il rapporto (fattore di miglioramento) tra il sistema in presenza di cache e in assenza di cache per l'esecuzione di questo programma.

Cache	4-set associativa
	8k parole
	tempo di accesso in lettura pari ad 1T
Blocco	64 parole
Ram	4M parole
	tempo di accesso in lettura pari a 10T
Algoritmo	LRU
Array	8320 parole

Essendo la cache 4-set associativa, essa ha 2^4 blocchi per set, quindi 16 blocchi per set.
La cache contiene 8k parole, quindi 8192 parole e 8 set che chiameremo set0,set1,...,set7.
L'array contiene 8320 parole e siccome in cache ne possono risiedere massimo 8192, si ha che 128 parole (corrispondenti a 2 blocchi) restano fuori dalla cache e dovranno essere caricate sovrascrivedo altri blocchi.

La scansione dell'array viene effettuata all'interno di un ciclo che viene eseguito 5 volte.Il tempo di trasferimento dalla cache alla CPU dipende soltanto dalla dimensione dell'array e della veelocità di accesso, quindi è indipendente dalla dimensione della cache:
8320*5*1T=41600T (1) è il tempo totale di trasferimento dalla cache alla CPU dell'array.
Calcoliamo ora il numero di accessi alla Ram per caricare la cache.
1^ ciclo 130 blocchi dalla Ram alla cache (la cache è inizialmente vuota). Utilizzando l'algoritmo LRU i blocchi 128 e 129 sostituiscono rispettivamente i blocchi 0 e 1 che occupano il set0 e il set1.
2^ - 5^ ciclo I blocchi 0 e 1 dovranno essere ricaricati dalla Ram, quindi sostituiranno i blocchi 8 e 9 del set0 rispettivamente set1.In questo modo tutti i blocchi del set0 e set1 veranno sostituiti con blocchi prelevati dalla Ram.In totale verranno fatti 16*2 accessi alla Ram per ogni ciclo.I set2,...,set7 rimangono inalterati.
Trasferimento di un blocco dalla Ram alla Cache: 64 (parole) * 10T (tempo di accesso in Ram)
(130+32*4)*64*10T=165120T (2) è il tempo di trasferimento dalla Ram alla cache.

Da cui si ottiene il tempo totale per l'esecuzione del programma

-IN PRESENZA DI CACHE
(1)+(2)=206720
-IN ASSENZA DI CACHE

8320 * 5 * 10T = 416000T

In presenza di Cache l'esecuzione del programma è circa 2 volte più veloce.
In questo caso l'algoritmo LRU opera sempre la scelta peggiore.Se avessimo usato l'algoritmo FIFO il numero di "cache miss" sarebbe stato, durante i cicli 2^ - 5^ , di 4 per ogni ciclo, quindi si otterebbe un fattore di miglioramento, tra il sistema in presenza di cache e in assenza di cache, maggiore di 3.

Esercizio 4 *Prima parte*
Rispondere alla seguente domanda riportando la motivazione della risposta.
• Una CPU con una pipeline a 2 stadi viene sostituita con una CPU con una pipeline a 4 stadi. Se il tempo totale di esecuzione di una singola istruzione è rimasto invariato, qual è il minimo ed il massimo incremento delle prestazioni che si può attendere?

Una pipeline ad n stadi è in grado di aumentare le prestazioni idealmente di n volte e di raggiungere lo scopo di riuscire ad eseguire una istruzione a ciclo di clock. Quindi se da una pipeline a due stadi si passa ad una a quattro stadi, essa può aumentare idealmente le prestazioni di due volte. Il minimo incremento delle prestazioni che si può ottenere è di una volta.

L'aumento degli stadi nella pipeline aumenta anche la probabilità di stallo della CPU. In condizione di istruzione di salto, nel caso in cui l'algoritmo di predizione dei salti fallisca, dato che più istruzioni vengono eseguite in parallelo, si annulla il vantaggio della pipeline con la perdita delle istruzioni fino a quel momento erroneamente elaborate, è quindi possibile che non si ottenga alcun incremento in quanto aumentando gli stadi della pipeline aumentano le condizioni di criticità, quindi si potrebbero formare delle bolle che manderebbero in stallo la CPU.

Esercizio 4 *Seconda parte*
• Si consideri una CPU con una pipeline a 4 stadi (F, D, O, W). Si riporti nel seguente diagramma, per ogni istruzione, lo stadio della pipeline coinvolto in *ogni istante di clock. Si ipotizzi la pipeline vuota al tempo 1.*

20 Clock/istruzione	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***
addl %eax, %ebx	F	D	E	W							
movl ind, %ecx		F	D	E	W						
subl %ebx, %ecx			F	D	D	D	E	W			
jz loop						F	D	D	D	E	W

Esercizio 4 *Terza parte*

• Descrivere il meccanismo e l'utilità della *predizione dei salti*.

I salti interrompono il flusso della pipeline e per minimizzare il numero di interruzioni è necessario uno schema di *brach prediction*.
I salti sono molto frequenti, in media uno ogni 6 istruzioni; nei processori superscalari che eseguono anche 4 istruzioni per ciclo di clock, la *predizione dei salti* è molto importante.
Molti schemi di *brach prediction* hanno un algoritmo che tiene traccia del comportamento di un dato salto l'ultima volta che è tato eseguito.
Infatti e un'istruzione l'ultima volta che è stata eseguita ha fatto un salto, si farà l'ipotesi che il salto venga preso nuovamente.
La pipeline ora contiene un'istruzione di salto condizionato e altre istruzioni successive, ma non si sa ancora se tali istruzioni serviranno o meno. Se il salto verrà preso, l'esecuzione del programma potrà continuare tranquillamente, se non sarà preso tutte le istruzioni caricate nella pipeline dovranno essere eliminate.

La *predizione dei salti* si divide in due: quella statica e quella dinamica.

Predizione STATICA:	è realizzata dal compilatore. La parola di codice operativo dell'istruzione indica all'unità di prelievo delle istruzioni se deve prevedere che il salto sia effettuato o meno. Il risultato della predizione è lo stesso ogni volta che si incontra una data istruzione di salto.
Predizione DINAMICA:	L'hardware del processore determina la probabilità che un salto venga eseguito ogni volta che si incontra una tale istruzione. Ciò può essere ottenuto mantenendo traccia del risultato della decisione di salto l'ultima volta che tale istruzione è stata eseguita e ipotizzando che la decisione nella presente istanza possa essere la stessa.

APPELLO 26/06/1999

Esercizio 3 *Prima parte*
3) Si consideri un sistema a memoria virtuale con spazio logico di 4G parole, una memoria fisica di 32M parole e dimensione delle pagine di 16K parole.
• *Determinare il numero di bit che definiscono:*
Lunghezza indirizzo logico:
Di cui per Num. Pag. Logica
per lo spiazzamento
Lunghezza indirizzo fisico
di cui per Num. Pag. Fisica
per lo spiazzamento

Spazio logico	4G parole
Memoria fisica	32M parole
Dimensione pagine	16k parole

Lunghezza indirizzo logico:	32
Di cui per Num. Pag. Logica:	18
Per lo spiazzamento:	14
Lunghezza indirizzo fisico:	25
Di cui per Num. Pag. Logica:	11
Per lo spiazzamento:	14

Esercizio 3 *Seconda parte*
• Nella seguente tabella sono riportati alcuni valori del parametro R (numero di pagine residenti per processo); sapendo che il Sistema Operativo occupa permanentemente 448 pagine e che sono stati creati 22 processi, indicare per ogni valore di R il numero di processi in stato di "fuori memoria" ossia che non possono avere tutte le pagine in memoria.

Dato **R** il numero di pagine residenti per processo, si ha:

Sistema Operativo	occupa 448 pagine permanentemente
Numero Processi	22

dim memoria fisica / dim pagina = numero totale di pagine
$2^{25} / 2^{14} = 2^{11} = 2048$ pagine

2048 - 448 = 1600 pagine libere

R	**40**	**80**	**160**	**320**	**800**
Numero di processi fuori memoria	0	2	12	17	20

Esercizio 3 *Terza parte*

• Specificare come è stato calcolato il valore per R = 160:

I processi in totale sono 22 e il numero di pagine residenti per processo 160, quindi il numero di pagine necessarie per far rimanere i processi in memoria sarebbe 22 * 160 = 3520.
Il numero di pagine disponibili è però 1600.
E' quindi evidente che i processi i quali possono restare attivi in memoria sono 10 (in quanto 10 * 160 = 1600) e per cui quelli fuori memoria sono 12 (22 - 10 = 12).

Esercizio 4

Elencare le micro istruzioni relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un solo BUS, che l'istruzione sia composta da una sola parola, che (%EAX) rappresenti un metodo di indirizzamento indiretto a registro e che l'indirizzo di salto della procedura sia relativo al PC:
CALL (%EAX)

1 bus
salto relativo
1. PC_{OUT}, MAR_{IN}, READ, CLEAR Y, SET CARRY-IN, ADD, Z_{IN}
2. WMFC, Z_{OUT}, PC_{IN}
3. MDR_{OUT}, IR_{IN}
4. ESP_{OUT}, CLEAR Y, SET CARRY-IN, SUB, Z_{IN}
5. Z_{OUT},ESP_{IN},MAR_{IN}
6. PC_{OUT}, MDR_{IN}, WRITE
7. WMFC
8. EAX_{OUT}, MAR_{IN}, READ
9. WMFC,PC_{OUT},Y_{IN}
10. MDR_{OUT}, ADD, Z_{IN}
11. Z_{OUT}, PC_{IN}, END

Commenti:
1.Fase di fetch, viene scaricato ed incrementato il PC.
2.Fase di fetch, il valore PC+1 viene memorizzato in PC.
3.Fase di fetch, l'indirizzo dell'istruzione da eseguire viene passato all'IR.
4.Viene decrementato il valore dell'ESP (stack pointer),inizio dell'operazione di PUSH nella pila di PC.
5.ESP viene caricato nella MAR per la fase di scrittura.
6.PC viene scritto nella cella di memoria puntata da ESP.
7.Attesa scrittura.
8.Il valore di EAX viene passato al MAR come indirizzo di memoria in quanto il salto è relativo.
9.Attesa lettura,PC viene caricato nel registro Y.
10.Dal MDR esce il valore puntato di EAX che viene sommato al PC.
11.L'indirizzo reale dell'istruzione da eseguire viene passato al PC.Termine dell'istruzione.

APPELLO 7/09/1999

Esercizio 3 *Prima parte*
Si consideri una memoria cache 4-set associativa della dimensione di 32 Kbyte con 1024 byte per blocco. La cache è collegata ad una memoria di 1Mbyte indirizzabile per byte. Definire le dimensioni ed il significato delle parti dell'indirizzo della cache e dell'indirizzo della RAM.

Cache	4-set associativa
	32 kbyte
Blocco	1 kbyte
Ram	1 Mbyte

Dimensioni dell'*indirizzo* necessario ad indirizzare tutta la memoria Ram:
dim. Ram 1M byte = 2^{20} quindi si hanno 20 bit per l'indirizzo

Dimensione del campo *parola*:
blocco = 1 kbyte = 2^{10} quindi si hanno 10 bit per la parola

Dimensione del campo *set*:
dim. Cache / dim. Blocco = 32k/1kbyte = $2^{15}/2^{10} = 2^5$
Questo risultato va ora diviso per $2^{n^{\wedge} set}$, quindi: $2^5/2^4 = 2^1$ da cui si ha 1 bit per il set

Dimensione del campo *etichetta*:
bit indirizzo della Ram - bit parola - bit set = Dimensione del campo etichetta
quindi
20 - 10 - 1 = 9

ETICHETTA	***SET***	***PAROLA***
9 bit	1 bit	10 bit

Dimensioni dell'indirizzo per la Cache:
32 kbyte = 2^{15} quindi si hanno 15 bit per l'indirizzo

Le dimensioni dei campi *parola* e *set* restano uguali, quindi si riduce la dimensione del campo *etichetta* che diventano:
15 - 10 - 1 = 4 bit

ETICHETTA	***SET***	***PAROLA***
4 bit	1 bit	10 bit

Esercizio 3 *Seconda parte*

• Quali sono i vantaggi e gli svantaggi delle memoria completamente associative rispetto alle memoria non associative.

Le cache associative sono, idealmente, migliori rispetto le cache non associative. Le cache associative non hanno infatti locazioni prestabilite ove scrivere determinati blocchi della memoria RAM, quindi prima di riscrivere un blocco della cache si occupano tutti i rimanenti blocchi liberi. Questo influisce notevolmente nelle prestazioni in quanto qualora i blocchi in cache vengano richiamati saranno con maggiore probabilità immediatamente a disposizione così da minimizzare l'evenienza di eseguire ulteriori accessi alla memoria tramite il BUS, con relativa perdita in tempo di esecuzione.

Tuttavia si evince subito il lato negativo: per le memorie completamente associative è necessario un algoritmo per scrivere i blocchi della RAM nei blocchi liberi o, in caso di cache già piena, nei blocchi meno utilizzati. Inoltre il maggior numero di porte logiche necessarie per la realizzazione delle cache completamente associative comporta una crescita dei costi di produzione, in quanto il controllo dei blocchi avviene in parallelo.

Esercizio 3 *Terza e Quarta parte*

• Quali sono le motivazioni che fanno preferire la realizzazione di unità di controllo cablate rispetto a quelle microprogrammate.

• Definire lo schema di un controllore cablato indicando i segnali utilizzati e la funzione dei blocchi presenti.

Le unità di controllo cablate sono pilotate da un segnale di clock. Ogni stato del contatore corrisponde ad uno dei passi dell'istruzione in esecuzione. Il tempo di clock minimo è il t_{MAX} di propagazione della rete combinatoria nel caso peggiore, quindi le unità di controllo cablate vantano una buona velocità.

Le unità di controllo microprogrammate sono invece molto più lente, per prelevare le istruzioni successive da eseguire devono continuamente accedere alla memoria.

Schema del controllore cablato:

I segnali di controllo richiesti sono univocamente determinati da:
- contenuto del contatore dei passi di controllo
- contenuto del registro delle istruzioni
- contenuto del codice di condizione e di altri flag di stato che rappresentano gli stati di diverse parti della CPU e delle linee di controllo legate ad esse.

Inoltre per ogni istruzione caricata in IR, una sola delle linee di uscita del decodificatore delle istruzioni è posta a a 1, le altre sono poste a 0.

Esercizio 4

Elencare le micro istruzioni relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un solo BUS, che l'istruzione sia composta da una sola parola e che (%EAX) rappresenti un metodo di indirizzamento indiretto a registro (usare solamente le righe necessarie):

CALL (%EAX)

1 bus

salto relativo

1. PC_{OUT}, MAR_{IN}, READ, CLEAR Y, SET CARRY-IN, ADD, Z_{IN}
2. WMFC, Z_{OUT}, PC_{IN}
3. MDR_{OUT}, IR_{IN}
4. ESP_{OUT}, CLEAR Y, SET CARRY-IN, SUB, Z_{IN}
5. Z_{OUT},ESP_{IN},MAR_{IN}
6. PC_{OUT}, MDR_{IN}, WRITE
7. WMFC
8. EAX_{OUT}, MAR_{IN}, READ
9. WMFC,PC_{OUT},Y_{IN}
10. MDR_{OUT}, ADD, Z_{IN}
11. Z_{OUT}, PC_{IN}, END

Commenti:

1.Fase di fetch, viene scaricato ed incrementato il PC.
2.Fase di fetch, il valore PC+1 viene memorizzato in PC.
3.Fase di fetch, l'indirizzo dell'istruzione da eseguire viene passato all'IR.
4.Viene decrementato il valore dell'ESP (stack pointer),inizio dell'operazione di PUSH nella pila del PC.
5.ESP viene caricato nella MAR per la fase di scrittura.
6.PC viene scritto nella cella di memoria puntata da ESP.
7.Attesa scrittura.
8.Il valore di EAX viene passato al MAR come indirizzo di memoria in quanto il salto è relativo.
9.Attesa lettura,PC viene caricato nel registro Y.
10.Dal MDR esce il valore puntato di EAX che viene sommato al PC.
11.L'indirizzo reale dell'istruzione da eseguire viene passato al PC.Termine dell'istruzione.

APPELLO 22/09/1999

Esercizio 3

Si consideri un programma che accede in sequenza a tutti gli elementi di un array di 10000 parole (ogni elemento ha dimensione di una parola). Questa operazione di scansione è effettuata all'interno di un ciclo che viene eseguito 10 volte.

Si assuma che le dimensioni di una pagina siano di 1k parole. Il programma occupa due pagine ed il suo working set è composto da 10 pagine.

Quanti page fault avvengono durante l'esecuzione del programma considerando che nessuna pagina del programma è residente in memoria al momento della sua esecuzione?

l codice occupa due pagine.

Parte di dati: 10000 parole (dati) / 1024 parole (dimensione pagina) = 9.76
Quindi servono 10 pagine per caricare tutto il programma.

Codice 1	Codice 2	Dati 1	Dati 2	Dati 3	Dati 4	Dati 5	Dati 6	Dati 7	Dati 8

1^ ciclo: Considerando che la memoria non contiene nessun pagina del programma,avvengono 12 page fault (2 per il caricamento del programma e 10 per il caricamento dell'array. La mappa della memoria alla fine della scansione è:

Codice 1	Codice 2	Dati 9	Dati 10	Dati 3	Dati 4	Dati 5	Dati 6	Dati 7	Dati 8

2^ - 10^ ciclo utilizzando l'algoritmo LRU: Le pagine Dati 9 e Dati 10 hanno sostituito le pagine Dati 1 e Dati 2 nel ciclo precedente, pagine che sono necessarie nel secondo ciclo di scansione. Per caricare la memoria, utilizzando LRU, la pagina Dati 1 sostituisce la pagina Dati 3 e la pagina Dati 2 sostituisce la pagina Dati 4. In questo modo non si avrà mai una pagina Dati utile nella memoria.

2^ - 10^ ciclo utilizzando l'algoritmo FIFO: Nel ciclo di scansione 1^ la pagina Dati 9 ha sostituito la pagina Dati 8 e poi è stata sostituita dalla pagina Dati 10. La mappa della memoria, prima del secondo ciclo, è la seguente:

Codice 1	Codice 2	Dati 1	Dati 2	Dati 3	Dati 4	Dati 5	Dati 6	Dati 7	Dati 10

Con l'algoritmo FIFO l'unica pagina che viene ripetutamente cambiata è l'ultima pagina, in cui verranno caricate rispettivamente le pagine Dati 8,Dati 9, Dati 10.

Concludendo,il numero di page fault è:
1^ ciclo: 12 page fault,indipendentemente dall'algoritmo utilizzato.
2^-10^ ciclo utilizzando LRU: 10 page_fault*9 cicli=90 page fault.
2^-10^ ciclo utilizzando FIFO: 3 page_fault*9 cicli=27 page fault.

Possiamo dire che,nel caso in cui la dimensione dei dati eccede leggermente la dimensione della memoria allocata, l'algoritmo vincente è FIFO mentre l'algoritmo LRU falisce sempre.

Esercizio 4

• Si consideri una CPU con una pipeline a 4 stadi (F, D, O, W). Si riporti nel seguente diagramma, per ogni istruzione, lo stadio della pipeline coinvolto in ogni istante di clock. Si ipotizzi la pipeline vuota al tempo 1.

Clock/istruzione	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***
mull %eax, %ebx	F	D	E	W							
movl %ecx, ind		F	D	E	W						
cmpl %ebx, 0h			F	D	D	E	W				
jnz inizio					F	D	D	D	E	W	

APPELLO 10/12/1999

Esercizio 3 *Prima parte*

Si consideri una memoria cache 4-set associativa della dimensione di 16 Kbyte con 512 byte per blocco. La cache è collegata ad una memoria di 2Mbyte indirizzabile per byte.

• Definire le dimensioni ed il significato delle parti dell'indirizzo della cache e dell'indirizzo della RAM.

Cache 4-set associativa
16 kbyte
Blocco 512 byte
Ram 2 Mbyte

Dimensioni dell'*indirizzo* necessario ad indirizzare tutta la memoria Ram:
dim. Ram 2M byte = 2^{21} quindi si hanno 21 bit per l'indirizzo

Dimensione del campo *parola*:
blocco = 512 byte = 2^9 quindi si hanno 9 bit per la parola

Dimensione del campo *set*:
dim. Cache / dim. Blocco = 16kbyte/512byte = $2^{14}/2^9 = 2^5$
Questo risultato va ora diviso per $2^{n^{\wedge}\ set}$, quindi: $2^5/2^4 = 2^1$ da cui si ha 1 bit per il set

Dimensione del campo *etichetta*:
bit indirizzo della Ram - bit parola - bit set = Dimensione del campo etichetta
quindi
21 - 9 - 1 = 11

ETICHETTA	***SET***	***PAROLA***
11 bit	1 bit	9 bit

Dimensioni dell'indirizzo per la Cache:
16 kbyte = 2^{14} quindi si hanno 14 bit per l'indirizzo

Le dimensioni dei campi *parola* e *set* restano uguali, quindi si riduce la dimensione del campo *etichetta* che diventano:
14 - 9 - 1 = 4 bit

ETICHETTA	***SET***	***PAROLA***
4 bit	1 bit	9 bit

APPELLO E II PROVA PARZIALE 27/6/2000

Esercizio 1 *Prima parte*
Elencare le micro istruzioni relative al caricamento, decodifica ed esecuzione della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un solo BUS, che l'istruzione sia composta da una sola parola, che (%EAX) rappresenti un metodo di indirizzamento indiretto a registro e che il salto condizionato sia di tipo diretto:
JZ (%EAX)

1 bus
JZ (%EAX)
1. PC_{OUT}, MAR_{IN}, READ, CLEAR Y, SET CARRY-IN, ADD, Z_{IN}
2. WMFC, Z_{OUT}, PC_{IN}
3. MDR_{OUT}, IR_{IN}
4. if !ZERO END, EAX_{OUT}, MAR_{IN}, READ
5. WMFC, PC_{OUT}, Y_{IN}
6. MDR_{OUT}, ADD, Z_{IN}
7. Z_{OUT}, PC_{IN}, END

Commenti:
1. fase di fetch, viene scaricato ed incrementato il PC
2. fase di fetch, viene messo in PC il valore PC+1
3. fase di fetch, escono i dati che vanno messi nel registro delle istruzioni
4. se il risultato non è zero termine dell'istruzione, altrimenti il contenuto di EAX viene passato al MAR come un indirizzo e viene letto
5. attesa per l'interpretazione dell'indirizzo, il valore corrente del PC viene messo nella locazione Y
6. l'indirizzo prodotto da MDR viene sommato a quello del PC attuale
7. l'indirizzo della prossima istruzione da eseguire viene messo nel PC. Fine dell'istruzione.

Esercizio 1 *Seconda parte*
• Elencare le ottimizzazioni che devono essere eseguite sull'architettura di un calcolatore per raggiungere l'obiettivo di avere, per la maggioranza delle istruzioni, CPI=1

Ottimizzazioni per ottenere $CPI_{medio} = 1$
1. Architettura a tre bus
2. incremento PC separato
3. cache
4. no operazioni su memoria
5. cache dati separata dalla cache del programma
6. Alu in cui tutte le operazioni vengono eseguite in un ciclo di clock
7. pipeline

Esercizio 2
Si consideri una CPU dotata di memoria cache di 128K byte con 64 byte per blocco. Questa CPU è collegata ad una memoria RAM da 16M byte indirizzabile per byte.

- Definire le dimensioni dell'indirizzo necessario a indirizzare tutta la memoria RAM e definire le dimensioni dei campi PAROLA, BLOCCO (o SET) ed ETICHETTA in cui questo indirizzo può essere suddiviso. Definire queste dimensioni per una memoria cache di tipo:
- ad accesso diretto
- 2-set associativa
- completamente associativa

Cache	accesso diretto
	2-set associativa
	completamente associativa
	128 kbyte
Blocco	64 byte
Ram	16 Mbyte

Dimensioni dell'*indirizzo* necessario ad indirizzare tutta la memoria Ram:
dim. Ram 16M byte = 2^{24} quindi si hanno 24 bit per l'indirizzo

Accesso diretto
Dimensione del campo *parola*:
blocco = 64 byte = 2^6 quindi si hanno 6 bit per la parola

Dimensione del campo *blocco*:
dim. Cache / dim. Blocco = 128kbyte/64byte = $2^{17}/2^6 = 2^{11}$

Dimensione del campo *etichetta*:
bit indirizzo della Ram - bit parola - bit blocco = Dimensione del campo etichetta
quindi
24 - 11 - 6 = 7

ETICHETTA	***BLOCCO***	***PAROLA***
7 bit	11 bit	6 bit

2-set Associativa
Dimensione del campo *parola*:
blocco = 64 byte = 2^6 quindi si hanno 6 bit per la parola

Dimensione del campo *set*:
dim. Cache / dim. Blocco = 128kbyte/64byte = $2^{17}/2^6 = 2^{11}$
Questo risultato va ora diviso per $2^{n^{\wedge}\ set}$, quindi: $2^{11}/2^2 = 2^9$ da cui si hanno 9 bit per il set

Dimensione del campo *etichetta*:
bit indirizzo della Ram - bit parola - bit set = Dimensione del campo etichetta
quindi
24 - 6 - 9 = 9

ETICHETTA	***SET***	***PAROLA***
9 bit	9 bit	6 bit

Completamente Associativa
Dimensione del campo *parola*:
blocco = 64 byte = 2^6 quindi si hanno 6 bit per la parola

Il campo *set* non ha luogo di essere.

Dimensione del campo *etichetta*:
bit indirizzo della Ram - bit parola = Dimensione del campo etichetta
quindi
24 - 6 = 18

ETICHETTA	***PAROLA***
18 bit	6 bit

Esercizio 3
Si descriva il meccanismo di interrupt.

Un interrupt è un meccanismo che permette ad un avento esterno al calcolatore di provocare il trasferimento del controllo da un programma ad un altro.

1. Quando il processore riceve un interrupt prima termina l'esecuzione in corso, poi carica nel PC l'indirizzo della prima istruzione della procedura di risposta interrupt.
2. Quando si verifica un interrupt il contenuto corrente del PC (che punta all'istruzione i+1) dev'essere memorizzato in una locazione temporanea.
   Devono essere salvate anche tutte le informazioni che possono essere modificate durante la risposta interrupt.

Esercizio 4
Si consideri una CPU con una pipeline a 4 stadi (F, D, E, W). Si riporti nel seguente diagramma, per ogni istruzione, lo stadio della pipeline coinvolto in ogni istante di clock. Si ipotizzi la pipeline vuota al tempo 1 e che la condizione del salto sia falsa.

Clock/istruzione	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***
cmpl %eax, %ebx	F	D	E	W							
jz START		F	D	D	D	E	W				
subl %ebx, %ecx					F	D	D	D	E	W	
movl %edx, DATA								F	D	E	W

APPELLO 18/07/2000

Esercizio 4
Elencare le micro istruzioni (insieme dei segnali di controllo) relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un solo BUS, che l'istruzione sia composta da una sola parola, che (%EAX) rappresenti un metodo di indirizzamento indiretto a registro e che il salto condizionato sia di tipo diretto:
JZ (%EAX)+%SI

1 bus
JZ (%EAX) + %SI
1. PC_{OUT}, MAR_{IN}, READ, CLEAR Y, SET CARRY-IN, ADD, Z_{IN}
2. WMFC, Z_{OUT}, PC_{IN}
3. MDR_{OUT}, IR_{IN}
4. if !ZERO END, EAX_{OUT}, MAR_{IN}, READ
5. WMFC, SI_{OUT}, Y_{IN}
6. MDR_{OUT}, ADD, Z_{IN}
7. Z_{OUT}, Y_{IN}
8. PC_{OUT}, ADD, Z_{IN}
9. Z_{OUT}, PC_{IN}, END

Commenti:
1. fase di fetch, viene scaricato ed incrementato il PC
2. fase di fetch, viene messo in PC il valore PC+1
3. fase di fetch, escono i dati che vanno messi nel registro delle istruzioni
4. se il risultato non è zero termine dell'istruzione, altrimenti il contenuto di EAX viene passato al MAR come un indirizzo e viene letto
5. attesa per l'interpretazione dell'indirizzo, il valore di SI viene messo nella locazione Y
6. l'indirizzo prodotto da MDR viene sommato a SI e viene memorizzato in Z
7. Il valore contenuto in Z viene trasferito in Y
8. Il valore di Y viene sommato al valore attuale del PC, il tutto viene salvato in Z
9. l'indirizzo della prossima istruzione da eseguire viene messo nel PC. Fine dell'istruzione.

APPELLO 05/09/2000

Esercizio 4

Elencare le micro istruzioni (insieme dei segnali di controllo) relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia TRE BUS, che l'istruzione sia composta da una sola parola, che (%EAX) rappresenti un metodo di indirizzamento indietto a registro e che il salto sia di tipo relativo:

CALL (%EAX)

1. $PC_{OUT\ A}$, NOP, $MAR_{IN\ C}$, READ
2. $PC_{OUT\ A}$, SET CARRY-IN, ADD, $PC_{IN\ C}$, WMFC
3. $MDR_{OUT\ B}$, IR_{IN}
4. $ESP_{OUT\ A}$, SET CARRY-IN, SUB, $ESP_{IN\ C}$,$MAR_{IN\ C}$
5. $PC_{OUT\ A}$, NOP, $MDR_{IN\ C}$, WRITE
6. WMFC
7. $EAX_{OUT\ A}$, NOP, $MAR_{IN\ C}$, READ
8. WMFC
9. $MDR_{OUT\ A}$, $PC_{OUT\ B}$, ADD, $PC_{IN\ C}$, END.

Commenti:

1. fase di fetch, il valore corrente del PC viene passato nel registro MAR
2. fase di fetch, il valore del PC viene incrementato di uno
3. fase di fetch, viene passato al registro delle istruzioni l'indirizzo della prossima istruzione da eseguire
4. fase di decode, il valore dello stack pointer viene decrementato,salvato e passato al registro degli indirizzi.
5. il valore del PC precedentemente incrementato di uno passa nel *memory date register*
6. attesa scrittura del PC nello stack.
7. il contenuto del registro EAX viene passato al MAR per esser interpretato
8. attesa
9. il contenuto di EAX esce dal MDR come un indirizzo che viene sommato al PC in quanto il salto è relativo, il nuovo indirizzo dell'istruzione da eseguire viene quindi memorizzato nel PC affinchè l'esecuzione possa continuare.

APPELLO 19/09/2000

Esercizio 4
Si consideri una memoria cache completamente associativa della dimensione di 64 Kbyte con 1024 byte per blocco. La cache è collegata ad una memoria di 16 Mbyte.
Definire le dimensioni ed il significato delle parti dell'indirizzo della cache e dell'indirizzo della RAM.

Dimensioni dell'*indirizzo* necessario ad indirizzare tutta la memoria Ram:
dim. Ram 16M byte = 2^{24} quindi si hanno 24 bit per l'indirizzo

Dimensione del campo *parola*:
blocco = 1 kbyte = 2^{10} quindi si hanno 10 bit per la parola

Dimensione del campo *etichetta*:
bit indirizzo della Ram - bit parola = Dimensione del campo etichetta
quindi
24 - 10 = 14

ETICHETTA	***PAROLA***
14 bit	10 bit

Dimensioni dell'indirizzo per la Cache:
64 kbyte = 2^{16} quindi si hanno 16 bit per l'indirizzo

La dimensione del campo *parola* resta uguale, quindi si riduce la dimensione del campo *etichetta* che diventano:
16 -10 = 6 bit

ETICHETTA	***PAROLA***
6 bit	10 bit

APPELLO 25/07/2001

Esercizio 3

• Si assuma che un computer con memoria virtuale, strutturata a pagine di 4Kbyte, sia dotato di 64Mbyte di memoria fisica.
Si consideri un programma con un codice di 7.2Kbyte che accede ciclicamente in sequenza a tutti gli elementi di un array di 1000 record in cui ogni campo è composto da 2 numeri interi. Quale deve essere la dimensione del *working set* perché si abbiano dei *page fault* solamente nella fase di caricamento del programma con esecuzione del primo ciclo di accesso agli elementi dell'array? Quanti *page fault* si avrebbero durante l'esecuzione del programma ipotizzando che il *working set* abbia una pagina meno di quanto definito nel punto precedente e che il ciclo di accesso venga ripetuto 10 volte?

Dim pagina	4 kbyte
memoria fisica	64 Mbyte
codice	7,2 kbyte
array	1000 record di cui ogni campo contiene 2 numeri interi

Il codice occupa due pagine.
Parte di dati: 2 (numeri) * 4 (byte) * 1000 (record) = 8000 byte
8000 byte : 4 kbyte = 1,95
La parte di dati occupa quindi 2 pagine.

Codice 1	Codice 2	Dati 1	Dati 2

La dimensione del Working Set perché si abbiano dei page fault solamente nella fase di caricamento del programma con esecuzione del primo ciclo di accesso agli elementi dell'array è di 4 pagine.

Ipotizzando che il working set abbia una pagina in meno di quanto definito nel punto precedente e che il ciclo di accesso venga ripetuto 10 volte, il numero di page fault che si avrebbero durante l'esecuzione del programma è 22, infatti:

Codice 1	Codice 2	Dati 1

1^ ciclo	4 page fault
2^ - 10^ ciclo	2 page fault * 9 cicli = 18 page fault

totale: 4 + 18 = 22 page fault

APPELLO 03/09/2001

Esercizio 4

Elencare le micro istruzioni relative alla completa esecuzione della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un solo BUS, che l'istruzione sia composta da una sola parola e che (%EBX) rappresenti un metodo di indirizzamento indiretto a registro:

ADDL (%EBX), %EAX
1 bus
1. PC_{OUT}, MAR_{IN}, READ, CLEAR Y, SET CARRY-IN, ADD, Z_{IN}
2. WMFC, Z_{OUT}, PC_{IN}
3. MDR_{OUT}, IR_{IN}
4. EBX_{OUT}, MAR_{IN}, READ
5. WMFC, EAX_{OUT}, Y_{IN}
6. MDR_{OUT}, ADD, Z_{IN}
7. Z_{OUT}, EAX_{IN}, END

Commenti:
1.fase di fetch, viene scaricato ed incrementato il PC
2.fase di fetch, viene messo in PC il valore PC+1
3.fase di fetch, escono i dati che vanno messi nel registro delle istruzioni
4.il contenuto di EBX viene passato al MAR come un indirizzo e viene letto
5.attesa per l'interpretazione dell'indirizzo, il valore di EAX viene messo nella locazione Y
6.l'indirizzo prodotto da MDR viene sommato a quello contenuto in Y
7.la somma dei due rgistri viene messa nel registro EAX. Fine dell'istruzione.

APPELLO 12/12/2001

Esercizio 4

Elencare le micro istruzioni (insieme dei segnali di controllo) relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia tre BUS, che l'istruzione sia composta da una sola parola e che (%EAX) rappresenti un metodo di indirizzamento indiretto a registro:
CALL (%EAX)

3 bus
Salto relativo
1.$PC_{OUT\ A}$, NOP, $MAR_{IN\ C}$, READ
2.$PC_{OUT\ A}$, SET CARRY-IN, ADD, $PC_{IN\ C}$, WMFC
3.$MDR_{OUT\ B}$, IR_{IN}
4.$ESP_{OUT\ A}$, SET CARRY-IN, SUB, $ESP_{IN\ C}$,$MAR_{IN\ C}$
5.$PC_{OUT\ A}$, NOP, $MDR_{IN\ C}$, WRITE
6.WMFC
7.$EAX_{OUT\ A}$, NOP, $MAR_{IN\ C}$, READ
8.WMFC
9.$MDR_{OUT\ A}$, $PC_{OUT\ B}$, ADD, $PC_{IN\ C}$, END.

Commenti:

1.Il valore del PC viene passato nel MAR per essere interpretato
2.il valore del PC viene incrementato
3.il valore del PC precedentemente interpretato dal MAR viene passato al registro delle istruzioni
4.il valore dello stack pointer viene decrementato e passato nel MAR
5.il valore del PC esce e viene passato al bus C e poi in MDR
6.attesa scrittura del PC nella pila.
7.il valore di EAX viene passato al MAR per essere interpretato come un indirizzo
8.attesa per l'interpretazione dell'indirizzo da parte del MAR
9.l'indirizzo ottenuto viene sommato al valore corrente del PC ed il risultato viene inserito in PC, in questo modo il PC ora contiene l'indirizzo della prossima istruzione da eseguire. Fine.

II PROVA PARZIALE 11/6/2002

Esercizio 3
Si consideri una CPU con una pipeline a 4 stadi (F, D, E, W). Si riporti nel seguente diagramma, per ogni istruzione, lo stadio della pipeline coinvolto in ogni istante di clock. Si ipotizzi la pipeline vuota al tempo 1.

Clock / istruzione	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
loop: addl %eax, %ebx	F	D	E	W									
movl ind, %ecx		F	D	E	W								
subl %ebx, %ecx			F	D	D	D	W						
jz loop						F	D	D	E	W			
movl %ecx, ind								F	D	D	D	E	W

APPELLO 26/03/2002

Esercizio 2 *Prima parte*
Elencare le micro istruzioni (insieme dei segnali di controllo) relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia **tre** BUS, che l'istruzione sia composta da una sola parola e che (%EAX) rappresenti un metodo di indirizzamento indiretto a registro (usare solamente le righe necessarie):
JMP (%EAX)

3 bus
salto relativo
JMP (%EAX)
1.$PC_{OUT\ A}$, NOP, $MAR_{IN\ C}$, READ
2.$PC_{OUT\ A}$, SET CARRY-IN, ADD, $PC_{IN\ C}$, WMFC
3.$MDR_{OUT\ B}$, IR_{IN}
4.$EAX_{OUT\ A}$, NOP, MAR_{IN}, READ
5.WMFC
6.$MDR_{OUT\ A}$, NOP, PC_{IN}, END

Commenti:
1.fase di fetch, viene scaricato il valore del PC e trasferito al MAR per essere interpretato
2.fase di fetch, il valore corrente del PC viene incrementato di uno e il nuovo valore ottenuto PC+1 viene memorizzato in PC
3.fase di fetch, l'indirizzo elaborato dal MDR viene trasferito nell'IR
4.il valore di EAX viene passato al MAR per essere interpretato come un indirizzo
5.attesa dell'interpretazione
6.l'indirizzo ottenuto viene passato al PC in modo tale che quest'ultimo contenga l'indirizzo della prossima istruzione da eseguire. Fine.

Esercizio 2 *Seconda parte*

• Si consideri una memoria cache 2-set associativa della dimensione di 64 Kbyte con 1024 byte per blocco. La cache è collegata ad una memoria di 4Mbyte indirizzabile per byte. Definire le dimensioni ed il significato delle parti dell'indirizzo della cache e dell'indirizzo della RAM.

Cache	2-set associativa 64 kbyte
Blocco	1 Kbyte
Ram	4 Mbyte

Dimensioni dell'*indirizzo* necessario ad indirizzare tutta la memoria Ram:
dim. Ram 4M byte = 2^{22} quindi si hanno 22 bit per l'indirizzo

Dimensione del campo *parola*:
blocco = 1 kbyte = 2^{10} quindi si hanno 10 bit per la parola

Dimensione del campo *set*:
dim. Cache / dim. Blocco = 64kbyte/1kbyte = 2^6
Questo risultato va ora diviso per $2^{n^{\wedge}\ set}$, quindi: $2^6/2^2 = 2^4$ da cui si hanno 4 bit per il set

Dimensione del campo *etichetta*:
bit indirizzo della Ram - bit parola - bit set = Dimensione del campo etichetta
quindi
22 - 10 - 4 = 8

ETICHETTA	***SET***	***PAROLA***
8 bit	4 bit	10 bit

Dimensioni dell'indirizzo per la Cache:
64 kbyte = 2^{16} quindi si hanno 16 bit per l'indirizzo

Le dimensioni dei campi *parola* e *set* restano uguali, quindi si riduce la dimensione del campo *etichetta* che diventano:
16 - 4 -10 = 2 bit

ETICHETTA	***SET***	***PAROLA***
2 bit	4 bit	10 bit

APPELLO 25/6/2002

Esercizio 3 *Prima parte*
Elencare le micro istruzioni relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un solo BUS, che l'istruzione sia composta da una sola parola, che (%EAX) rappresenti un metodo di indirizzamento indiretto a registro e che l'indirizzo di salto della procedura sia relativo al PC:
JNZ (%EAX)

1 bus
salto relativo
JNZ (%EAX)
1. PC_{OUT}, MAR_{IN}, READ, CLEAR Y, SET CARRY-IN, ADD, Z_{IN}
2. WMFC, Z_{OUT}, PC_{IN}
3. MDR_{OUT}, IR_{IN}
4. if ZERO END, EAX_{OUT}, MAR_{IN}, READ
5. WMFC, PC_{OUT}, Y_{IN}
6. MDR_{OUT}, ADD, Z_{IN}
7. Z_{OUT}, PC_{IN}, END

Commenti:

1. fase di fetch, viene scaricato ed incrementato il PC
2. fase di fetch, viene messo in PC il valore PC+1
3. fase di fetch, escono i dati che vanno messi nel registro delle istruzioni
4. se il risultato è zero termine dell'istruzione, altrimenti il contenuto di EAX viene passato al MAR come un indirizzo e viene letto
5. attesa per l'interpretazione dell'indirizzo, il valore corrente del PC viene messo nella locazione Y
6. l'indirizzo prodotto da MDR viene sommato a quello del PC attuale
7. l'indirizzo della prossima istruzione da eseguire viene messo nel PC. Fine dell'istruzione.

Esercizio 3 *Seconda parte*
• Quali sono le motivazioni che spingono i microprocessori moderni a essere dotati di una pipeline.

Lo scopo di adottare le pipeline nei processori moderni è quello di ridurre il più possibile il CPI_{medio} e di ottenere $CPI_{medio} = 1$.
Le pipeline sono delle strutture interne alla CPU; esse sono dotate di più stadi (come ad esempio Fetch, Decode, Execute e WriteBack) e permettono di ridurre il CPI aumentando l'IPS.
Per avere dei netti miglioramenti in quantità di tempo è indispensabile cercare di fare entrare la CPU in stallo il meno possibile, di tenerla quindi sempre a regime.
Per fare questo, un meccanismo indispensabile è la *predizione dei salti* (*branch prediction*).
Teoricamente se una pipeline ha *n* stadi, aumenta l'efficienza di *n* volte.
In realtà non è così poiché aumenta la probabilità di dipendenza fra i dati, motivo per cui i processori moderni cercano di avere molti stadi ma non eccessivi, ed un sistema di *branch prediction* il più affidabile possibile.

APPELLO 16/07/2002

Esercizio 4

Elencare le micro istruzioni relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un solo BUS, che l'istruzione sia composta da una sola parola, che (%Exx) rappresenti un metodo di indirizzamento indiretto a registro e che l'indirizzo di salto della procedura sia assoluto:
JNZ (%EAX + %EBX)

1 bus
salto relativo
JNZ (%EAX + %EBX)
1. PC_{OUT}, MAR_{IN}, READ, CLEAR Y, SET CARRY-IN, ADD, Z_{IN}
2. WMFC, Z_{OUT}, PC_{IN}
3. MDR_{OUT}, IR_{IN}
4. if ZERO END, EAX_{OUT}, Y_{IN}
5. EBX_{OUT}, ADD, Z_{IN}
6. Z_{OUT}, MAR_{IN}, READ
7. WMFC
8. MDR_{OUT}, PC_{IN}, END

Commenti:

1. fase di fetch, viene scaricato ed incrementato il PC
2. fase di fetch, viene messo in PC il valore PC+1
3. fase di fetch, escono i dati che vanno messi nel registro delle istruzioni
4. se il risultato è zero termine dell'istruzione, altrimenti il contenuto di EAX viene passato in Y
5. EBX viene sommato al valore di EAX ed il risultato viene memorizzato in Z
6. la somma ottenuta viene passata al MAR come un indirizzo da interpretare
7. attesa
8. l'indirizzo della prossima istruzione da eseguire (quindi la somma di EAX ed EBX) viene messo nel PC. Fine dell'istruzione.

APPELLO 06/09/2002

Esercizio 4

Elencare e commentare le micro istruzioni relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un solo BUS, che l'istruzione sia composta da una sola parola, che (%Exx) rappresenti un metodo di indirizzamento indiretto a registro e che l'indirizzo di salto della procedura sia assoluto:
CALL (%EAX + %EBX)

1 bus
CALL (%EAX + %EBX)
1. PC_{OUT}, MAR_{IN}, READ, CLEAR Y, SET CARRY-IN, ADD, Z_{IN}
2. WMFC, Z_{OUT}, PC_{IN}
3. MDR_{OUT}, IR_{IN}
4. ESP_{OUT}, CLEAR Y, SET CARRY-IN, SUB, Z_{IN}
5. Z_{OUT}, ESP_{IN}, MAR_{IN}
6. PC_{OUT},MDR_{IN},WRITE
7. WMFC
8. EAX_{OUT},Y_{IN}
9. EBX_{OUT},ADD, Z_{IN}
10. Z_{OUT}, MAR_{IN}, READ
11. WMFC
12. MDR_{OUT},PC_{IN}, END.

Commenti:
1.Fase di fetch, viene scaricato ed incrementato il PC
2.Fase di fetch, il valore PC+1 viene memorizzato in PC
3.Fase di fetch, l'indirizzo dell'istruzione da eseguire viene passato all'IR
4.Viene decrementato il valore dell'ESP (stack pointer)
5.Il valore dell'ESP decrementato viene passato al registro degli indirizzi
6.Il valore di PC viene passato al registro dei dati per essere scritto nella memoria
7.Attesa scrittura
8.Esce il valore di EAX
9.Il valore di EBX viene sommato a EAX
10.EAX+EBX viene passato al registro degli indirizzi per la lettura
11.Attesa
12.L'indirizzo reale dell'istruzione da eseguire viene passato al PC in quanto il salto è assoluto. Termine dell'istruzione.

APPELLO 20/09/2002

Esercizio 2
Si consideri una memoria cache 4-set associativa della dimensione di 32 Kbyte con 1024 byte per blocco. La cache è collegata ad una memoria di 4Mbyte indirizzabile per byte.
• Definire le dimensioni ed il significato delle parti dell'indirizzo della cache e dell'indirizzo della RAM.

Cache	4-set associativa 32 kbyte
Blocco	1024 byte
Ram	4 Mbyte

Dimensioni dell'*indirizzo* necessario ad indirizzare tutta la memoria Ram:
dim. Ram 4M byte = 2^{22} quindi si hanno 22 bit per l'indirizzo

Dimensione del campo *parola*:
blocco = 1 kbyte = 2^{10} quindi si hanno 10 bit per la parola

Dimensione del campo *set*:
dim. Cache / dim. Blocco = 32kbyte/1kbyte = 2^5
Questo risultato va ora diviso per $2^{n\text{^ set}}$, quindi: $2^5/2^4 = 2^1$ da cui si ha 1 bit per il set

Dimensione del campo *etichetta*:
bit indirizzo della Ram - bit parola - bit set = Dimensione del campo etichetta
quindi
22 - 10 - 1 = 11

ETICHETTA	***SET***	***PAROLA***
11 bit	1 bit	10 bit

Dimensioni dell'indirizzo per la Cache:
32 kbyte = 2^{15} quindi si hanno 15 bit per l'indirizzo

Le dimensioni dei campi *parola* e *set* restano uguali, quindi si riduce la dimensione del campo *etichetta* che diventano:
15 - 1 -10 = 4 bit

ETICHETTA	***SET***	***PAROLA***
4 bit	1 bit	10 bit

Esercizio 3
Elencare le micro istruzioni (insieme dei segnali di controllo) relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un solo BUS, che l'istruzione sia composta da una sola parola, che (%EAX) rappresenti un metodo di indirizzamento indiretto a registro e che il salto sia relativo:
CALL (%EAX)+%SI

1 bus
salto relativo
CALL (%EAX) + %SI
1. PC_{OUT}, MAR_{IN}, READ, CLEAR Y, SET CARRY-IN, ADD, Z_{IN}
2. WMFC, Z_{OUT}, PC_{IN}
3. MDR_{OUT}, IR_{IN}
4. ESP_{OUT}, MAR_{IN}, CLEAR Y, SET CARRY-IN, SUB, Z_{IN}
5. PC_{OUT}, MDR_{IN}, WRITE
6. WMFC, Z_{OUT}, ESP_{IN}
7. EAX_{OUT}, MAR_{IN}, READ
8. WMFC, SI_{OUT}, Y_{IN}
9. MDR_{OUT}, ADD, Z_{IN}
10. Z_{OUT}, Y_{IN}
11. PC_{OUT}, ADD, Z_{IN}
12. Z_{OUT}, PC_{IN}, END

Commenti:
1. Fase di fetch, viene scaricato ed incrementato il PC
2. Fase di fetch, il valore PC+1 viene memorizzato in PC
3. Fase di fetch, l'indirizzo dell'istruzione da eseguire viene passato all'IR
4. Viene decrementato il valore dell'ESP (stack pointer) in modo tale da lasciare una cella libera ed il valore ESP-1 viene memorizzato in Z
5. Il valore del PC viene passato al MDR per essere scritto
6. Attesa. Viene salvato nell'ESP il valore ESP-1
7. Esce il valore contenuto in EAX e viene passato al MAR per essere interpretato come un indirizzo
8. Attesa. Il valore di SI viene passato in Y
9. Esce quindi dal MDR l'indirizzo della prossima istruzione da eseguire e viene sommato a SI (che è contenuto in Y)
10,11. Il contenuto di Z viene sommato a PC in quanto il salto è relativo.
12. Caricamento dell'indirizzo di salto in PC.Termine dell'istruzione.

APPELLO 13/12/2002

Esercizio 2
Si consideri una CPU dotata di memoria cache 2-associativa di 8K parole con 64 parole per blocco. Questa CPU è collegata ad una memoria RAM da 8M parole. Definire le dimensioni dell'indirizzo necessario a indirizzare tutta la memoria RAM e definire le dimensioni dei campi PAROLA, BLOCCO ed ETICHETTA in cui questo indirizzo può essere suddiviso. Motivare la risposta con un opportuno schema.

Cache	2-set associativa 8k parole
Blocco	64 parole
Ram	8M parole

Dimensioni dell'*indirizzo* necessario ad indirizzare tutta la memoria Ram:
dim. Ram 8M parole = 2^{23} quindi si hanno 23 bit per l'indirizzo

Dimensione del campo *parola*:
blocco = 64 parole = 2^6 quindi si hanno 6 bit per la parola

Dimensione del campo *set*:
dim. Cache / dim. Blocco = $2^{13}/2^6 = 2^7$
Questo risultato va ora diviso per $2^{n^{\wedge}\ set}$, quindi: $2^7/2^2 = 2^5$ da cui si hanno 5 bit per il set

Dimensione del campo *etichetta*:
bit indirizzo della Ram - bit parola - bit set = Dimensione del campo etichetta
quindi
23 - 5 - 6 = 12

ETICHETTA	***SET***	***PAROLA***
12 bit	5 bit	6 bit

Esercizio 4
Elencare le micro istruzioni relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia tre BUS, che l'istruzione sia composta da una sola parola, che (%Exx) rappresenti un metodo di indirizzamento indiretto a registro e che l'indirizzo di salto della procedura sia assoluto:
JNZ (%EAX) + %EBX

3 bus
Salto assoluto
JNZ (%EAX) + %EBX
1.$PC_{OUT\ A}$, NOP, $MAR_{IN\ C}$, READ
2.$PC_{OUT\ A}$, SET CARRY-IN, ADD, $PC_{IN\ C}$, WMFC
3.$MDR_{OUT\ B}$, IR_{IN}
4.if ZERO END, $EAX_{OUT\ A}$, NOP, $MAR_{IN\ C}$, READ
5.WMFC
6.$MDR_{OUT\ A}$, $EBX_{OUT\ B}$, ADD, $PC_{IN\ C}$, END

Commenti:

1.Fase di fetch, il valore corrente del PC viene passato nel registro MAR
2.Fase di fetch, il valore del PC viene incrementato di uno
3.Fase di fetch, viene passato al registro delle istruzioni l'indirizzo della prossima istruzione da eseguire
4.Se il risultato è zero: termine dell'istruzione, altrimenti: il valore di EAX viene passato al MAR per essere interpretato come un indirizzo
5.Attesa
6.L'elaborazione prodotta dal MDR viene sommata al valore di EBX e il risultato è l'indirizzo della prossima istruzione da eseguire, che quindi viene memorizzata nel PC. Fine.

II PROVA PARZIALE 12/06/2003

Esercizio 1 *Prima parte*
Si consideri una memoria cache 3-set associativa della dimensione di 64 Kbyte con 1024 byte per blocco. La cache è collegata ad una memoria di 4 Mbyte indirizzabile per byte. Definire le dimensioni ed il significato delle parti dell'indirizzo della cache e dell'indirizzo della RAM.

Cache	3-set associativa
	64 kbyte
Blocco	1024 byte
Ram	4 Mbyte

Dimensioni dell'*indirizzo* necessario ad indirizzare tutta la memoria Ram:
dim. Ram 4M byte = 2^{22} quindi si hanno 22 bit per l'indirizzo

Dimensione del campo *parola*:
blocco = 1 kbyte = 2^{10} quindi si hanno 10 bit per la parola

Dimensione del campo *set*:
dim. Cache / dim. Blocco = 64kbyte/1kbyte = $2^{16}/2^{10} = 2^6$
Questo risultato va ora diviso per $2^{n^{\wedge}\ set}$, quindi: $2^6/2^3 = 2^3$ da cui si hanno 3 bit per il set

Dimensione del campo *etichetta*:
bit indirizzo della Ram - bit parola - bit set = Dimensione del campo etichetta
quindi
22 - 10 - 3 = 9

ETICHETTA	***SET***	***PAROLA***
9 bit	3 bit	10 bit

Dimensioni dell'indirizzo per la Cache:
64 kbyte = 2^{16} quindi si hanno 16 bit per l'indirizzo

Le dimensioni dei campi *parola* e *set* restano uguali, quindi si riduce la dimensione del campo *etichetta* che diventano:
16 - 3 -10 = 3 bit

ETICHETTA	***SET***	***PAROLA***
3 bit	3 bit	10 bit

Esercizio 1 *Seconda parte*
Quali sono i vantaggi e gli svantaggi delle memorie completamente associative rispetto alle memoria non associative.

1 Memorie *completamente associative*:

Vantaggi	***Svantaggi***
Non ci sono blocchi di Cache predefiniti ove memorizzare i blocchi di Ram, quindi la Cache è utilizzabile per intero ed i conflitti di blocco compaiono solo quando la Cache è piena. Questo implica massima libertà nella scelta di dove posizionare in Cache il blocco di Ram.	E' necessario un algoritmo di ricerca per trovare i blocchi ricercati all'interno della Cache.
Lo spazio della Cache è usato in modo molto efficiente.	Costo elevato.

2 Memorie *non associative*:

Vantaggi	***Svantaggi***
Facile da realizzarsi.	Non molto flessibile.
Unico blocco in Cache ove memorizzare uno specifico blocco della Ram, quindi è sufficiente confrontare i bit più significativi dell'indirizzo con l'etichetta associata al blocco della Cache per sapere se il blocco è presente in Cache oppure no.	

Esercizio 2
Elencare e **commentare** le micro istruzioni relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un solo BUS, che l'istruzione sia composta da una sola parola, che (%EAX) rappresenti un metodo di indirizzamento indiretto a registro e che il salto sia di tipo relativo:
CALL (%EAX)

CALL (%EAX)
1 bus
salto relativo
1. PC_{OUT}, MAR_{IN}, READ, CLEAR Y, SET CARRY-IN, ADD, Z_{IN}
2. WMFC, Z_{OUT}, PC_{IN}
3. MDR_{OUT}, IR_{IN}
4. ESP_{OUT}, CLEAR Y, SET CARRY-IN, SUB, Z_{IN}
5. Z_{OUT},ESP_{IN},MAR_{IN}
6. PC_{OUT}, MDR_{IN}, WRITE
7. WMFC
8. EAX_{OUT}, MAR_{IN}, READ
9. WMFC,PC_{OUT},Y_{IN}
10. MDR_{OUT}, ADD, Z_{IN}
11. Z_{OUT}, PC_{IN}, END

Commenti:
1.Fase di fetch, viene scaricato ed incrementato il PC.
2.Fase di fetch, il valore PC+1 viene memorizzato in PC.
3.Fase di fetch, l'indirizzo dell'istruzione da eseguire viene passato all'IR.
4.Viene decrementato il valore dell'ESP (stack pointer),inizio dell'operazione di PUSH nella pila del PC.
5.ESP viene caricato nella MAR per la fase di scrittura.
6.PC viene scritto nella cella di memoria puntata da ESP.
7.Attesa scrittura.
8.Il valore di EAX viene passato al MAR come indirizzo di memoria in quanto il salto è relativo.
9.Attesa lettura,PC viene caricato nel registro Y.
10.Dal MDR esce il valore puntato di EAX che viene sommato al PC.
11.L'indirizzo reale dell'istruzione da eseguire viene passato al PC.Termine dell'istruzione.

APPELLO 01/07/2003

Esercizio 4 *Prima parte*
Quali sono le ottimizzazioni per una architettura basata sul modello di Von Neumann necessarie per raggiungere l'obiettivo di un CPI medio uguale a 1?

Ottimizzazioni per ottenere $CPI_{medio} = 1$

3 Architettura a tre bus
4 incremento PC separato
5 cache
6 no operazioni su memoria
7 cache dati separata dalla cache del programma
8 Alu in cui tutte le operazioni vengono eseguite in un ciclo di clock
9 pipeline

Esercizio 4 *Seconda parte*
• Si consideri una CPU con una pipeline a 4 stadi (F, D, E, W). Si riporti nel seguente diagramma, per ogni istruzione, lo stadio della pipeline coinvolto in ogni istante di clock. Si ipotizzi che la pipeline sia vuota al tempo 1 e che `jz` faccia riferimento all'istruzione `subl`.

b**Clock / istruzione**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
addl %eax, %ebx	F	D	E	W									
movl %ebx, %ecx		F	D	D	D	E	W						
subl %eax, %ecx					F	D	D	D	E	W			
jz loop								F	D	D	D	E	W

Esercizio 4 *Terza parte*
• Quali sono i vantaggi di una *cache* a indirizzamento diretto?

Nella cache ad accesso diretto il set è composto da un solo blocco e ogni blocco della RAM ha un unico posto in cache ove essere memorizzato.
I bit più significativi dell'indirizzo vengono confrontati con l'etichetta associata al blocco della Cache: se coincidono la parola cercata si trova in quel blocco della Cache, altrimenti non c'è altra posizione ove cercarla e bisogna caricarla dalla Ram.
Il metodo di indirizzamento diretto è facile da realizzarsi (anche se non è molto flessibile).

APPELLO 11/07/2003

Esercizio 4

Si consideri una CPU dotata di memoria cache 3-set associativa di 8K parole con 64 parole per blocco. Questa CPU è collegata ad una memoria RAM da 8M parole.

• Definire le dimensioni dell'indirizzo necessario a indirizzare tutta la memoria RAM e definire le dimensioni dei campi PAROLA, BLOCCO ed ETICHETTA in cui questo indirizzo può essere suddiviso. Motivare la risposta con un opportuno schema.

Cache	3-set associativa 8k parole
Blocco	64 parole
Ram	8M parole

Dimensioni dell'*indirizzo* necessario ad indirizzare tutta la memoria Ram:
dim. Ram 8M parole = 2^{23} quindi si hanno 23 bit per l'indirizzo

Dimensione del campo *parola*:
blocco = 64 parole = 2^6 quindi si hanno 6 bit per la parola

Dimensione del campo *set*:
dim. Cache / dim. Blocco = 8k parole/64 parole = $2^{13}/2^6 = 2^7$
Questo risultato va ora diviso per $2^{n^{\wedge}\ set}$, quindi: $2^7/2^3 = 2^4$ da cui si hanno 3 bit per il set

Dimensione del campo *etichetta*:
bit indirizzo della Ram - bit parola - bit set = Dimensione del campo etichetta
quindi
23 - 4 - 6 = 13

ETICHETTA	***SET***	***PAROLA***
13 bit	4 bit	6 bit

APPELLO 02/09/2003

Esercizio 3
Elencare e commentare le micro istruzioni relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel 80386 AT&T), assumendo che la CPU abbia un solo BUS, che l'istruzione sia composta da una sola parola, che (%Exx) rappresenti un metodo di indirizzamento indiretto a registro e che l'indirizzo di salto della procedura sia assoluto:
CALL (%EAX + %EBX)

1 bus
CALL (%EAX + %EBX)
1. PC_{OUT}, MAR_{IN}, READ, CLEAR Y, SET CARRY-IN, ADD, Z_{IN}
2. WMFC, Z_{OUT}, PC_{IN}
3. MDR_{OUT}, IR_{IN}
4. ESP_{OUT}, CLEAR Y, SET CARRY-IN, SUB, Z_{IN}
5. Z_{OUT}, ESP_{IN}, MAR_{IN}
6. PC_{OUT}, MDR_{IN}, WRITE
7. WMFC
8. EAX_{OUT}, Y_{IN}
9. EBX_{OUT}, ADD, Z_{IN}
10. Z_{OUT}, MAR_{IN}, READ
11. WMFC
12. MDR_{OUT}, PC_{IN}, END.

Commenti:
1.Fase di fetch, viene scaricato ed incrementato il PC
2.Fase di fetch, il valore PC+1 viene memorizzato in PC
3.Fase di fetch, l'indirizzo dell'istruzione da eseguire viene passato all'IR
4.Viene decrementato il valore dell'ESP (stack pointer)
5.Il valore dell'ESP decrementato viene passato al registro degli indirizzi
6.Il valore di PC viene passato al registro dei dati per essere scritto nella memoria
7.Attesa scrittura
8.Esce il valore di EAX
9.Il valore di EBX viene sommato a EAX
10.EAX+EBX viene passato al registro degli indirizzi per la lettura
11.Attesa
12.L'indirizzo reale dell'istruzione da eseguire viene passato al PC in quanto il salto è assoluto. Termine dell'istruzione.

APPELLO 16/09/2003

Esercizio 3
Elencare e commentare le micro istruzioni relative alla completa esecuzione (caricamento, decodifica, esecuzione) della seguente istruzione assembler (Intel

80386 AT&T), assumendo che la CPU abbia un solo BUS, che l'istruzione sia composta da una sola parola, che (%Exx) rappresenti un metodo di indirizzamento indiretto a registro, che $4 sia un indirizzamento immediato e che l'indirizzo di salto della procedura sia assoluto (usare solamente le righe necessarie e commentare ogni istruzione):
CALL (%EAX + $4)

1 bus
salto assoluto
CALL (%EAX + $4)
1. PC_{OUT}, MAR_{IN}, READ, CLEAR Y, SET CARRY-IN, ADD, Z_{IN}
2. WMFC, Z_{OUT}, PC_{IN}
3. MDR_{OUT}, IR_{IN}
4. ESP_{OUT}, CLEAR Y, SET CARRY-IN, SUB, Z_{IN}
5. Z_{OUT}, ESP_{IN},MAR_{IN}
6. PC_{OUT}, MDR_{IN}, WRITE
7. WMFC, EAX_{OUT}, Y_{IN}
8. $OFFSET(IR)_{OUT}$,ADD, Z_{IN}
9. Z_{OUT},MAR_{IN}, READ
10. WMFC
11. MDR_{OUT}, PC_{IN}, END

Commenti:
1. Fase di fetch, viene scaricato ed incrementato il PC
2. Fase di fetch, il valore PC+1 viene memorizzato in PC
3. Fase di fetch, l'indirizzo dell'istruzione da eseguire viene passato all'IR
4. Viene decrementato il valore dell'ESP (stack pointer) in modo tale da puntare a una cella vuota
5. Il valore di ESP passa nel registro degli indirizzi.
6. Il PC passa nel registro dei dati e viene eseguita l'operazione di scrittura.
7. Attesa scrittura nello stack,mentre il valore di EAX viene caricato nel registro buffer Y
8. L'offset di IR che contiene il valore 4 viene sommato a EAX
9. Il risultato della somma passa al registro degli indirizzi,in quanto il metodo di indirizzamento è indiretto.
10. Attesa lettura
11. Il valore letto passa nel PC,in quanto il salto è assoluto.