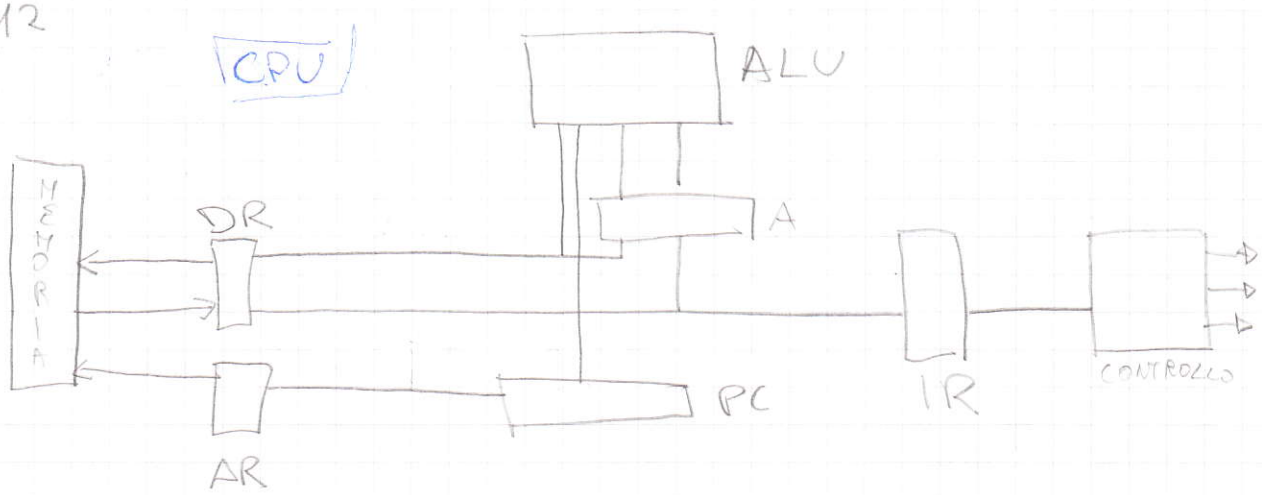
FOND. INFO B

3/06/2005

Disegnare la struttura di una semplice CPU e indicare le operaz. elementari a svolgere tutte le fasi (fetch e esecuzione) di una istruzione assembly che esegua una operazione di LOAD

LEZ 10 s 12



LOAD A, (mem1)

Fetch

sempre uguale

- AR ← PC
- DR ← M(AR)
- IR ← DR (codice op)
- PC ← PC+1

Esecuzione

- AR ← DR
- DR ← AR (mem)
- A ← DR

STORE (mem1), A

Esecuzione

- AR ← DR (indirizz)
- DR ← A
- (mem) AR ← DR (scrit.)

SUB A, #22

Esecuzione

A ← A−DR

ADD A, #22

A ← A+DR

CALL 3345$_H$ → salvataggio e poi caricamento

Esecuz.

1ª ipotesi

cod. oper.	3345$_H$

SP ← SP − 1

AR ← SP

R_{TEMP} ← DR (indirizzo)

DR ← PC

M(AR) ← DR

PC ← R_{TEMP}

PUSH A

Esec.

SP ← SP − 1

AR ← SP

DR ← A

M(AR) ← DR

RETURN

AR ← SP

DR ← M(AR)

PC ← DR

SP ← SP + 1

Fondamenti di Informatica B 10/06/05

Domande del compito

3) Definire la funzione svolta da un multiplexer e disegnarne lo schema a livello logico di un caso semplice.

è un circuito che (bit)
Il multi- seleziona uno tra più dati in ingresso e li
invia verso una destinazione comune.
Col multi- è possibile realizzare ogni tipo di funzione Booleana
Se abbiamo 1 funz. con n variabili il nostro multiplexer avrà 2^n ingressi da 1 bit ed n ingressi per la selezione codificata

seriale-parallelo

2) ~~~ decodificatore ~~~

Decoder o demultiplexer consiste in un circuito combinatorio che, in funzione del valore degli n ingressi, pone a 1 una sola delle 2^n variabili di uscita

parallelo-seriale

8) Registro scorrimento generalizzato

Funzioni svolte:

ingresso parallelo
" seriale
uscita parallelo
" seriale
abilitazione scorrimento
scorrimento a destra
" " sinistra

10) modi indirizzamento assembler

L'indirizzo serve per conoscere la posizione del dato richiesto.
Immediato - dato contenuto nel codice macchina
es: MOV A,#8
Diretto - il codice m. contiene l'indirizzo del dato
es: MOV A,X
Indiretto - il c.m. contiene l'indirizzo della posizione in memoria dell'indirizzo del dato richiesto
es: MOV A,(x)

20) Stack pointer

? Altro modo di indirizzamento: es. MOV A,(x)+
registro che a fine esecuzione viene incrementato

26) Sistema RTL. + esempio

RTL, o Register Transfer Level, è il livello superiore a quello logico nella gerarchia di analisi di un sistema di elaborazione. È composto da circuiti combinatori o sequenziali che elaborano e immagazzinano informazioni binarie raggruppate in parole (o vettori).
~~Nel RTL~~ Non esistono simboli standard universalmente per i circuiti a livello di registro.

Es. moltiplicatore binario

$$\begin{cases} P_i \Leftarrow P_i + X_{n-i} \cdot y_H \\ P_{i+1} \Leftarrow P_i \cdot 2^{-1} \end{cases}$$

ripetere da 0 a 6

29) principio Località

prevede la modalità di accesso ai dati:

Se al tempo t si accede all'indirizzo x è molto probabile che l'indirizzo $x+\Delta x$ sia richiesto tra t e $t+\Delta t$

30) Gerarchia di memoria

La [illegible] è la soluzione a cui si ricorre per soddisfare la coesistenza di un sistema di memoria, di solito: tempi d'accesso minimi, capacità max, ad un costo min...

32) Tempo di accesso alla memoria

t_s seek time + t_L latency time — $t_L = (2r)^{-1}$

testina che si muove tra le tracce — testina posizionata sul dato → dipende da r_c (vel. di rotazione)

39) Memorie

Interna: x ms cache
Principali: RAM
Secondarie: HD - CD - DVD

26) Sistema RTL. + esempio

RTL, o Register Transfer Level, è il livello superiore a quello logico nella gerarchia di analisi di un sistema di elaborazione. È composto da circuiti combinatori o sequenziali che elaborano e immagazzinano informazioni binarie raggruppate in parole (o word o byte) ~~del RTL~~. Non esistono simboli adottati universalmente per i circuiti a livello di registro.

Es. moltiplicatore binario

ripetere da 0 a 6:

$$\begin{cases} P_i \Leftarrow P_i + X_{N-i} \cdot y_N \\ P_{i+1} \Leftarrow P_i \cdot 2^{-1} \end{cases}$$

29) principio Località

prevede la modalità di accesso ai dati:

se al tempo t si accede all'indirizzo x è molto probabile che l'indirizzo $x+\Delta x$ sia richiesto tra t e $t+\Delta t$

30) Gerarchia di memoria

La [illegible] è la soluzione a cui si ricorre per soddisfare le esigenze di un sistema di memoria, che sono: tempi d'accesso minimi, capacità max [illegible] al contempo costo min...

32) Tempo di accesso alla memoria

t_s seek time + t_L latency time — $t_L = (2r)^{-1}$

testina che si muove fino a 6 tracce — testina posizionata sul dato → dipende da r_c (vel. di rotazione)

39) Memorie

Interna: x ms — cache

Principale: RAM

Secondaria: HD - CD - DVD

FONDAMENTI D'INFORMATICA B
Domande II° Compitino

DOMANDA 1

1. Descrivere la funzione svolta da una unità di controllo e la sua interazione con il percorso dati in un sitema digitale.
2. Definire la funzione svolta da un decodificatore o decoder e disegnarne lo schema a livello logico di un caso semplice.
3. Definire la funzione svolta da un multiplexer e disegnarne lo schema a livello logico di un caso semplice.
4. Quale componente consente di realizzare la conversione serie-parallelo dei dati ? Spiegare con un esempio il comportamento del modulo.
5. Quale componente consente di realizzare la conversione parallelo-serie dei dati ? Spiegare con un esempio il comportamento del modulo.
6. ~~Definire la funzione svolta da un multiplexer e disegnarne lo schema a livello logico in un caso semplice.~~
7. ~~Quale componente consente di realizzare la conversione serie-parallelo dei dati ? Spiegare con un esempio il comportamento del modulo.~~
8. Descrivere le caratteristiche e le funzioni svolte da un registro a scorrimento generalizzato.
9. ~~Descrivere le caratteristiche e le funzioni svolte da un registro a scorrimento generalizzato.~~
10. ~~Descrivere le caratteristiche e le funzioni svolte da un registro a scorrimento generalizzato.~~
11. L'architettura di Von Neuman. Caratteristiche generali (10 punti).
12. ~~Descrivere le caratteristiche e le funzioni svolte da un registro a scorrimento generalizzato.~~
13. ~~Definire la funzione svolta da un decodificatore o decoder e disegnarne lo schema a livello logico di un caso semplice.~~

DOMANDA 2

14. Modi di indirizzamento dei dati nel linguaggio assembler.
15. Disegnare la struttura di una semplice CPU e indicare le operazioni elementari necessarie a svolgere una istruzione assembler che esegua una operazione di sottrazione tra un dato memorizzato nell'accumulatore e un dato definito in modo immediato e infine memorizzi il risultato nell'accumulatore: **SUB A,#22.**
16. Disegnare la struttura di una semplice CPU e indicare le operazioni elementari necessarie a svolgere una istruzione assembler che esegua una operazione di somma tra un dato memorizzato nell'accumulatore e un dato definito in modo immediato e infine memorizzi il risultato nell'accumulatore: **ADD A,#22.**
17. Disegnare la struttura di una semplice CPU e indicare le operazioni elementari necessarie a svolgere tutte le fasi (fetch ed esecuzione) di una istruzione assembly che esegua una operazione di LOAD **LOAD A,(mem1)**
18. Disegnare la struttura di una semplice CPU e indicare le operazioni elementari necessarie a svolgere tutte le fasi (fetch ed esecuzione) di una istruzione assembly che esegua una operazione di STORE **STORE (mem1),A**
19. Disegnare la struttura di una semplice CPU e indicare le operazioni elementari necessarie a svolgere tutte le fasi (fetch ed esecuzione) di una istruzione assembly che esegua una operazione di CALL **CALL 3345_H.**

20. Mettere in evidenza il ruolo svolto dallo Stack Pointer nella esecuzione della istruzione.
21. Disegnare la struttura di una semplice CPU e indicare le operazioni elementari necessarie a svolgere tutte le fasi (fetch ed esecuzione) di una istruzione assembly che esegua una operazione di CALL

CALL 1345_H.

22. Mettere in evidenza il ruolo svolto dallo Stack Pointer nella esecuzione della istruzione.

23. Disegnare la struttura di una semplice CPU e indicare le operazioni elementari necessarie a svolgere una istruzione assembler che esegua una operazione di prodotto logico tra un dato memorizzato nell'accumulatore e un dato definito in modo immediato e infine memorizzi il risultato nell'accumulatore: **AND A, #22.**

24. I sottoprogrammi. Descrivere in dettaglio il meccanismo che rende possibili le chiamate a sottoprogrammi e l'annidamento dei sottoprogrammi stessi facendo riferimento a un frammento di codice assembler.

25. I sottoprogrammi. Descrivere in dettaglio il meccanismo che rende possibili le chiamate a sottoprogrammi e l'annidamento dei sottoprogrammi stessi facendo riferimento a un frammento di codice assembler.

26. Un sistema a livello RTL. Illustrare con un esempio la tecnica descrittiva (10 punti).

27. Disegnare la struttura di una semplice CPU e indicare le operazioni elementari necessarie a svolgere tutte le fasi (fetch ed esecuzione) di una istruzione assembly che esegua una operazione di OR esclusivo tra due dati memorizzati in registri (A e B) e ponga il risultato nel registro A.

XOR A,B.

28. Disegnare la struttura di una semplice CPU e indicare le operazioni elementari necessarie a svolgere tutte le fasi (fetch ed esecuzione) di una istruzione assembly che esegua una operazione di somma tra due dati memorizzati in registri (A e B) e ponga il risultato nel registro A.

ADD A,B.

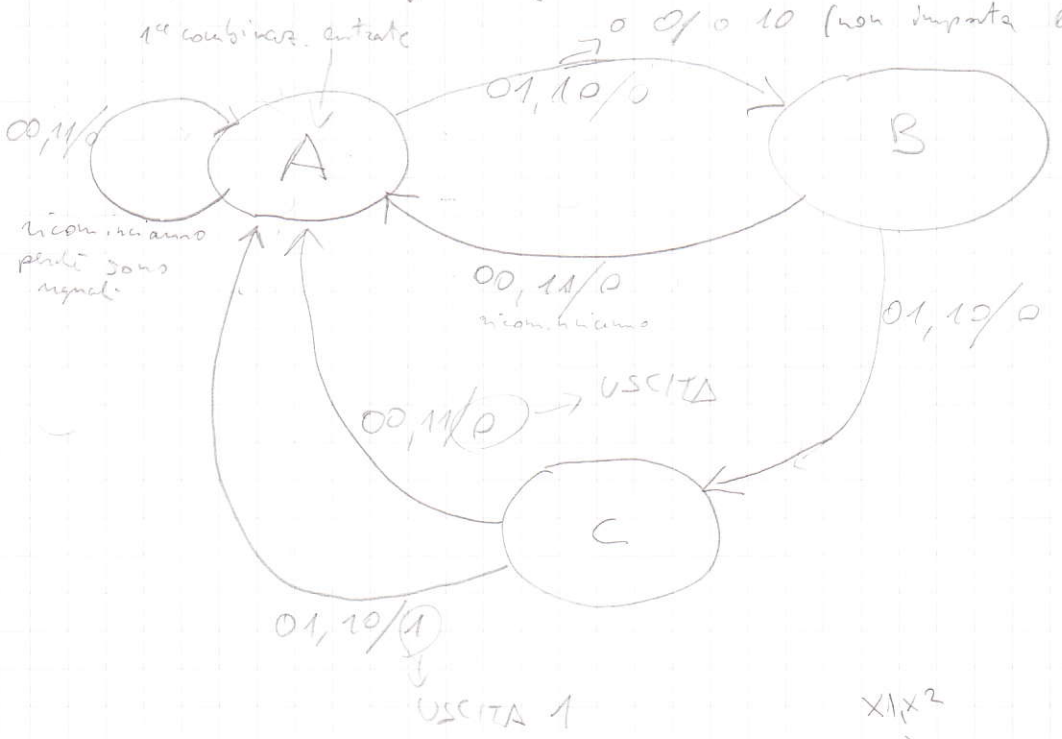
DOMANDA 3

29. Il principio di località. Illustrare il concetto di base e l'importanza che riveste in un sistema di calcolo.
30. Cosa si intende per gerarchia di memoria ?
31. Cosa si intende per principio di località ?
32. Tempo di accesso alla memoria. Discutere i parametri che determinano il tempo di accesso alla memoria nel caso di una struttura gerarchica.
33. Cosa si intende per unità di controllo in un sistema digitale? Che ruolo svolge ?
34. Tempo di accesso alla memoria. Discutere i parametri che determinano il tempo di accesso alla memoria nel caso di una struttura gerarchica.
35. Cosa si intende per unità di controllo in un sistema digitale? Che ruolo svolge ?
36. Tempo di accesso alla memoria. Definire i parametri significativi anche nel caso di una struttura gerarchica.
37. Tempo di accesso alla memoria. Definire i parametri significativi anche nel caso di una struttura gerarchica.
38. Tempo di accesso alla memoria. Definire i parametri significativi anche nel caso di una struttura gerarchica.
39. Descrivere molto brevemente le principali tecnologie delle memorie usate nei sistemi di calcolo e indicarne le caratteristiche in termini di dimensioni e di tempi di accesso (10 punti).
40. Principio di località. Illustrare l'importanza di tale principio nei sistemi di elaborazione moderni.
41. Gerarchia di memoria. Illustrare l'importanza di tale soluzione tecnologica nei moderni sistemi di calcolo.

INFO B

1)

Progettare una rete sequenziale sincrona che ha il compito di verificare l'andamento di 2 segnali di ingresso x_1 e x_2. La rete dispone di una uscita Z, che assume valore 1 in corrispondenza del 3° intervallo di tempo consecutivo, in cui la coppia di segnali di ingresso assume valore diverso (cioè i 2 segnali sono diversi). Subito dopo questo intervallo la rete riprende il conteggio. Progettare il circuito mediante FF-JK. → sincroni
SR - senza clock



	00	01	11	10
00 ≡ A	A,0	B,0	A,0	B,0
01 ≡ B	A,0	C,0	A,0	C,0
10 ≡ C	A,0	A,1	A,0	A,1

X_1X_2

	00	01	11	10
00	00,0	01,0	00,0	01,0
01	00,0	10,0	00,0	10,0
10	00,0	00,1	00,0	00,1

X_1	X_2	F_1	F_2	F_{1+}	F_{2+}	J_1	K_1	J_2	K_2	Z
0	0	0 A	0	0	0	0	–	0	–	0
0	0	0 B	1	0	0	0	–	–	1	0
0	0	1 C	0	0	0	–	1	0	–	0
0	0	1 X	1	–	–	–	–	–	–	–
0	1	0	0	0	1	0	–	1	–	0
0	1	0	1	1	0	1	–	–	1	0
0	1	1	0	0	0	–	1	0	–	1
0	1	1	1	–	–	–	–	–	–	–
1	0	0	0	0	1					0
1	0	0	1	1	0					0
1	0	1	0	0	0	come				1
1	0	1	1	–	–					–
1	1	0	0	0	0					0
1	1	1	0	0	0	come				0
1	1	1	1	–	–					–

X_1X_2 entrate

stato futuro F_1F_2

uscita

$X_1X_2=01$
$F_1F_2=10$

come funziona FF - JK

clock	entrate		uscita
CK	J	K	Q_{n+1}
0	-	-	Q_n
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	Q_n^2

tabella delle transizioni

presente	futuro		
Q_n	Q_{n+1}	J	K
0	0	0	-
0	1	1	-
1	0	-	1
1	1	-	0
0	-	-	-

ora continua la tabella di prima con J_1, K_1...

J_1

x_1x_2 / F_1F_2	00	01	11	10
00	0	0	0	0
01	0	1	0	1
11	-	-	-	-
10	-	-	-	-

raccolgo

$\bar{x}_2$ va in parte prendiamo come 1 e raccolgo con 1

J_1

Soluzione $J_1 = \bar{F}_2 \bar{x}_1 x_2 + \bar{F}_2 x_1 \bar{x}_2$

K_1

x_1x_2 / F_1F_2	00	01	11	10
00	-	-	-	-
01	-	-	-	-
11	-	-	-	-
10	1	1	1	1

raccoglimento maggiore possibile

K_1

potenze di 2: 1 2 4 8 16

raccoglimento quadrati

$K_1 = 1$ non cambia - cambierebbe se avessi raccolto x_2 a colonne

J_2

x_1x_2 / F_1F_2	00	01	11	10
00	0	1	0	1
01	-	-	-	-
11	-	-	-	-
10	0	0	0	0

$J_2 = \bar{F}_1 \bar{x}_1 x_2 + \bar{F}_1 x_1 \bar{x}_2$

F_2 cambia e non si tocca

K_2

x_1x_2 / F_1F_2	00	01	11	10
00	-	-	-	-
01	1	1	1	1
11	-	-	-	-
10	-	-	-	-

$K_2 = 1$