Architettura degli Elaboratori - Corso A

Prima Prova di Verifica Intermedia, a.a. 2004-05, 15 novembre 2004

Correzione

Queste note, oltre che documentare la soluzione della prova, hanno lo scopo di mostrare come organizzare un elaborato, ed in particolare come apportare le necessarie spiegazioni. Per ragioni didattiche, vengono commentate più soluzioni alternative allo stesso problema e vengono dati alcuni richiami sulla teoria; per queste ragioni, le spiegazioni qui introdotte sono in numero decisamente superiore rispetto a quelle che ogni singolo elaborato dovrebbe mediamente contenere.

Domanda 1

Una unità di elaborazione U è così definita:

i) possiede al suo interno due componenti logici memoria, A e B, entrambi di capacità 64K parole;

ii) riceve dall'unità U_0 messaggi (OP, J1, J2, X), dove OP è il codice operativo delle operazioni esterne, J1 e J2 sono indirizzi, e X è una parola;

iii) operazioni esterne:

1) copia il valore di X nella locazione di A di indirizzo J1;

2) copia il blocco di 256 parole consecutive di A, che inizia all'indirizzo J1, nel blocco di B che inizia all'indirizzo J2. In questa operazione esterna, è garantito che i valori di J1 e J2 inviati da U_0 siano multipli interi di 256;

3) invia all'unità U_1 il valore OUT uguale al numero di locazioni di A e B, *con lo stesso indirizzo*, aventi lo stesso contenuto.

Tutte le comunicazioni tra unità utilizzano interfacce a transizione di livello.

È noto il ritardo t_p di una porta logica con al massimo 8 ingressi. Una ALU ha ritardo uguale a $5t_p$. Le memorie A e B devono essere realizzate mediante memorie di capacità 2K parole, ognuna avente tempo di accesso uguale a $5t_p$. La durata dell'impulso di clock è uguale a t_p.

a) Scrivere il microprogramma. Spiegare le scelte effettuate, ed in particolare spiegare come sono state scelte le variabili di condizionamento.

b) Mostrare la Parte Operativa. Ricavarne la funzione delle uscite e la funzione di transizione dello stato interno, quest'ultima funzione relativamente solo al registro con il quale viene indirizzata la memoria B. Spiegare come sono state ricavate tali funzioni.

c) Ricavare la funzione di transizione dello stato interno della Parte Controllo e la funzione delle uscite della Parte Controllo, quest'ultima funzione relativamente solo alla variabile di controllo dell'indicatore di interfaccia d'ingresso usato per segnalare "messaggio pronto" inviato da U_0. Spiegare come sono state ricavate tali funzioni.

d) Valutare il ciclo di clock in funzione di t_p, e ricavare il tempo medio di elaborazione dell'operazione esterna 2).

Domanda 2

Mostrare la sintassi e spiegare il formato e la semantica delle seguenti istruzioni assembler Risc (Cap. V Dispensa):

a) addizione,

b) trasferimento da una locazione di memoria in un registro generale,

c) salto condizionato con predicato diadico e indirizzo di salto relativo al contatore istruzioni,

d) salto incondizionato con indirizzo di salto indiretto via registro.

Domanda 1

a) Microprogramma

La figura 1 mostra le interfacce dell'unità U. Ogni collegamento fa capo ad un registro di interfaccia; in particolare ai collegamenti RDY e ACK corrispondono indicatori di interfaccia a transizione di livello che verranno mostrati in seguito nello schema della Parte Operativa. Essendo le memorie A e B di capacità 64K, gli indirizzi J1 e J2 sono rappresentati su 16 bit. Il codice operativo OP è di due bit, OP_0 e OP_1, che faranno parte delle variabili di condizionamento assieme a RDY_0 e ACK_1. La codifica delle operazioni esterne è: *00* per la 1, *01* per la 2, *1 –* per la 3.

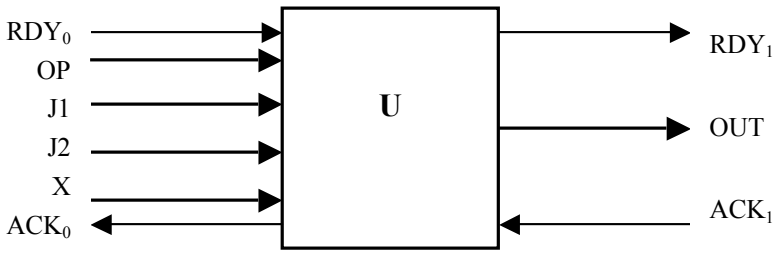

Fig. 1

Nel microprogramma dell'operazione esterna 2, che comporta un loop ripetuto 256 volte, occorre salvare gli indirizzi J1 e J2 in due registri, INDA e INDB, per indirizzare le memorie A e B rispettivamente.

Questi registri verranno usati con lo stesso scopo anche nel microprogramma dell'operazione esterna 3. Pur se in questa sarebbe sufficiente usare uno stesso registro per indirizzare sia A che B (ad esempio, INDA), conviene comunque usare gli stessi registri nell'operazione esterna 2 e nella 3, soprattutto per ridurre il numero di possibili sorgenti per l'indirizzamento delle memorie (numero che, in generale, contribuisce ad aumentare il tempo di accesso). Nessun vantaggio di otterrebbe ad introdurre un ulteriore registro di indirizzamento da usare solo nel microprogramma dell'operazione esterna 3. La soluzione di usare indirizzamento base+indice aumenterebbe ulteriormente il ritardo per l'accesso alle memorie.

Invece, nel microprogramma dell'operazione esterna 1, che è eseguibile in un singolo ciclo di clock, per i valori di J1 e X verranno usati direttamente i registri di ingresso corrispondenti.

Per il controllo del loop nei microprogrammi delle operazioni esterne 2 e 3 verrà fatto uso di un registro I di 17 bit, inizializzato rispettivamente ai valori 255 e 64K-1 (65535), e decrementato di uno ad ogni passo. La condizione di uscita dal loop corrisponde a $I_0 = 1$, con I_0 bit del segno, che costituisce una variabile di condizionamento. Alternativamente, si sarebbero potuti adottare due registri contatori, uno di 9 bit (operazione esterna 2) ed uno di 17 bit (operazione esterna 3), inizializzati e fatti variare secondo più soluzioni.

Nel microprogramma dell'operazione esterna 3, importante è la scelta della variabile di condizionamento per testare, ad ogni passo del loop, la condizione *A[INDA] = B[INDB].* Sono possibili almeno due soluzioni. La prima è del tipo:

i. … zero(A[INDA] – B[INDB]) → Z, … , i+1; …

i+1. (… Z … = … 0 …) Frase 0 ; (… Z … = … 1 …) Frase 1 ; …

con Z registro di 1 bit posto sull'uscita secondaria "zero" di una ALU.

La seconda soluzione è del tipo:

i. (… zero(A[INDA] – B[INDB]) … = … 0 …) Frase 0 ;

(… zero(A[INDA] – B[INDB]) … = … 1 …) Frase 1 ;

Si noti che, al posto della funzione *zero(A[INDA] – B[INDB])*, si sarebbe potuto usare qualunque altra funzione equivalente per il confronto tra due numeri, ad esempio *zero(A[INDA] ⊕ B[INDB])*: in ogni soluzione, avendo a che fare con numeri rappresentati su parola, occorre sempre usare una ALU, e quindi

tutte queste soluzioni hanno lo stesso impatto sul ritardo della funzione ω_{PO} (nel secondo caso) o della σ_{PO} (nel primo caso).

La seconda soluzione comporta un numero di cicli di clock complessivo uguale alla metà del numero di cicli di clock della prima, ed un ciclo di clock più lungo a causa del ritardo maggiore della funzione ω_{PO} (ritardo di stabilizzazione della ALU per valutare il predicato "zero"). Poiché presumibilmente il ciclo di clock della seconda soluzione sarà minore del doppio rispetto alla prima (condizione che, comunque, andrebbe verificata alla fine), il tempo medio di elaborazione della seconda soluzione sarà inferiore: verrà quindi scelta la seconda soluzione. Affinché questa soluzione sia corretta, cioè affinché la rete sequenziale PO risponda al modello matematico di Moore, occorre però verificare che il valore della variabile di condizionamento *zero(A[INDA] – B[INDB])* dipenda, *ad ogni ciclo di clock*, soltanto dallo stato interno della PO stessa e non da valori di variabili di controllo; in particolare, devono essere rispettate due condizioni:

1) venga usata una ALU capace di eseguire solo l'operazione di sottrazione ed i cui ingressi sia dati esclusivamente dalle uscite delle memorie A e B (oppure: usando una ALU multifunzione o con più ingressi, le variabili di controllo siano ottenute come uscite di registri),

2) le memorie A e B siano indirizzate esclusivamente da INDA e INDB, oppure che per l'operazione di lettura siano usati esclusivamente tali indirizzi (a meno che eventuali variabili di controllo di commutatori posti sull'indirizzo delle memorie siano ottenute come uscite di registri).

Oltre alla due soluzioni descritte, sono possibili ulteriori soluzioni che, con inizializzazioni opportune (e, talvolta, un po' macchinose) delle variabili, tendano ad ottimizzare sia il numero di cicli di clock che la lunghezza del ciclo di clock.

Con le scelte discusse finora, il microprogramma di U è il seguente:

0. (RDY_0, OP_0, OP_1 = 0 – –) nop, 0;

   (= 1 0 0) reset RDY_0, set ACK_0, X → A[J1], 0;

   (= 1 0 1) reset RDY_0, set ACK_0, J1 → INDA, J2 → INDB, 255 → I, 1;

   (= 1 1 –) reset RDY_0, set ACK_0, 0 → INDA, 0 → INDB, (64K-1) → I, 0 → C, 2

   // il registro C, di 17 bit, è il temporaneo per il risultato dell'operazione esterna 3 //

1. (I_0 = 0) A[INDA] → B[INDB], INDA + 1 → INDA, INDB + 1 → INDB, I – 1 → I, 1;

   (= 1) "nop", 0

   // questa "nop" è una solo abbreviazione: può essere eliminata fondendo la microistruzione 0 nella 1 //

2. (I_0, zero(A[INDA] – B[INDB]), ACK_1 = 0 0 –) INDA + 1 → INDA, INDB + 1 → INDB, I – 1 → I, 2;

   (= 0 1 –) C + 1 → C, INDA + 1 → INDA, INDB + 1 → INDB, I – 1 → I, 2;

   (= 1 – 0) nop, 2;

   (= 1 – 1) reset ACK_1, set RDY_1, C → OUT, 0

b) Parte Operativa

Lo schema della PO è mostrato nella figura 2. Per ricavare questo schema, notiamo che:

- le reti ALU sono in numero di cinque: quattro servono *per permettere il parallelismo* nella microoperazione della seconda frase della microistruzione 2 (tre incrementatori ed un decrementatore), la quinta (sottrattore) per disporre di una ALU indipendente allo scopo di soddisfare la condizione 1) relativamente ad una PO di Moore.

•

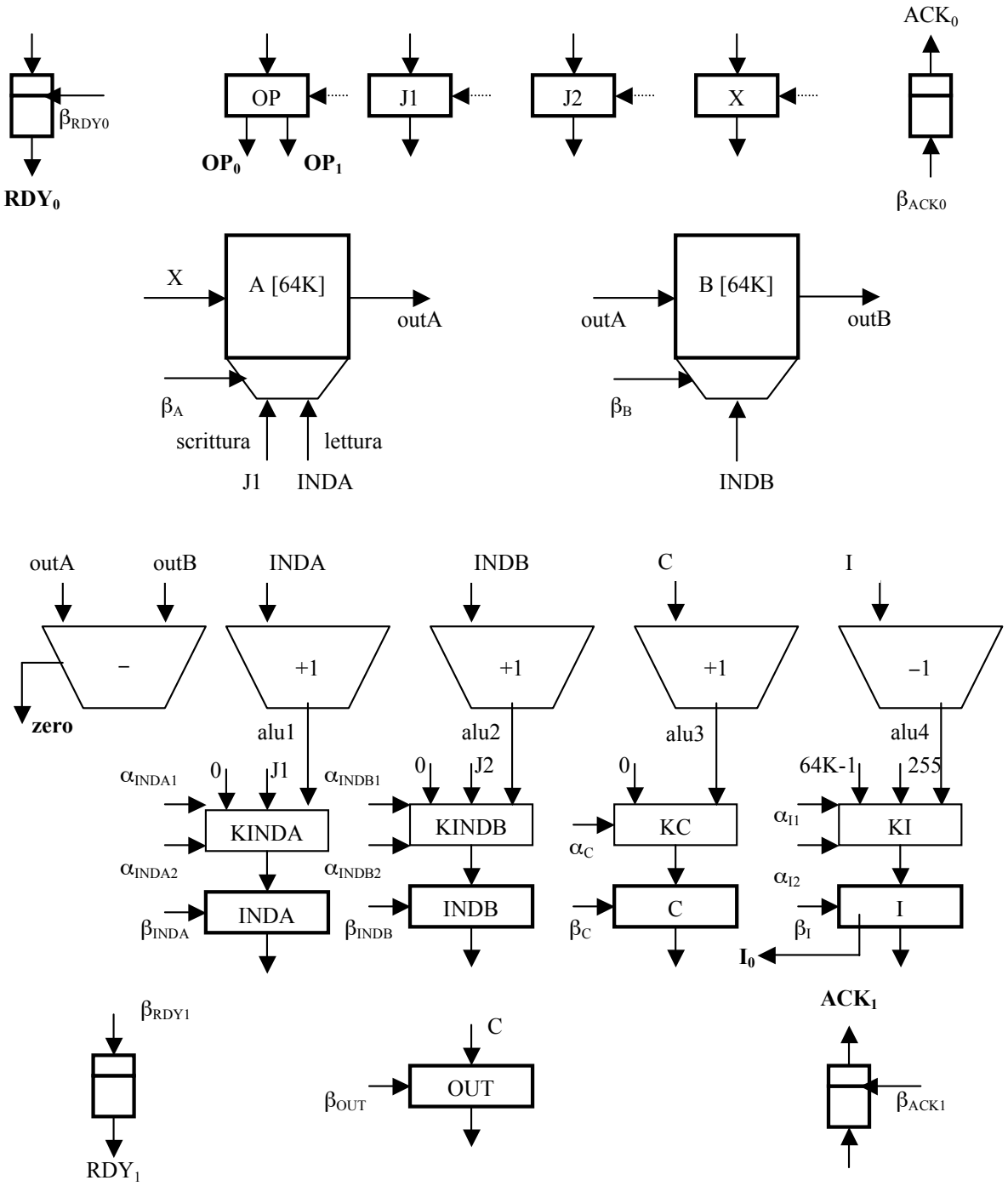

Fig. 2

È importante ribadire che le ALU utilizzate sono nel numero *minimo* per permettere il parallelismo presente nel microprogramma: *sarebbe invece un errore introdurre tante ALU per quante sono le operazioni aritmetico-logiche presenti nel microprogramma indipendentemente dal parallelismo esplicitato*. Inoltre, il fatto che nella PO di figura 2 non ci siano commutatori sugli ingressi delle ALU è, in questo caso, del tutto fortuito, e non va assolutamente considerata una regola;

- la memoria A è indirizzata in scrittura mediante il registro J1, in lettura mediante il registro INDA; per soddisfare la condizione 2) relativamente ad una PO di Moore, occorre infatti che tale memoria disponga di una logica separata per l'indirizzamento in scrittura (selezionatore controllato da J1) e per l'indirizzamento in lettura (commutatore controllato da INDA). Si noti che *non ha senso usare questa tecnica per un numero qualunque di logiche di indirizzamento distinte*; ad esempio (per l'unità U non è proponibile avere qualcosa del tipo: 3 logiche per l'indirizzamento in lettura e 2 in scrittura);

- i registri C, I, INDA, INDB hanno una struttura tra loro analoga;
- lo schema delle interfacce è quello standard con la soluzione a transizione di livello: gli indicatori di uscita (ACK_0, RDY_1) hanno un unico ingresso dato da una variabile di controllo per consentire l'operazione *set*, quelli di ingresso (RDY_0, ACK_1) hanno, oltre all'ingresso esterno, un ulteriore ingesso dato da una variabile di controllo per consentire l'operazione *reset*.

Funzione delle uscite della PO

Indicando con lettere *x* le variabili di condizionamento e con lettere *z* le uscite esterne, *dall'analisi della struttura della rete sequenziale PO* si ricavano le espressioni logiche di tutte le variabili di uscita in funzione dello stato interno della PO stessa, verificando che in tali espressioni non compiano variabili di controllo:

$x_0 = RDY_0$ $\qquad$ $z_0 = ACK_0$

$x_1 = OP_0$ $\qquad$ $z_1 = RDY_1$

$x_2 = OP_1$ $\qquad$ $z_2 = OUT$ (parola)

$x_3 = I_0$

$x_4 = zero(A[INDA] - B[INDB])$

$x_5 = ACK_1$

In particolare, dall'espressione di x_4 si verifica che, *con la realizzazione data,* PO è una rete sequenziale di Moore.

Funzione di transizione dello stato interno della PO

Limitatamente al registro INDB, occorre esprimere il valore che, ad ogni ciclo di clock, è assunto dall'ingresso di tale registro in funzione dello stato interno della PO e dello stato d'ingresso della PO (variabili di controllo). Si ha:

in_{INDB} = **when** $\beta_{INDB} = 1$ **do**

switch α_{INDB1} α_{INDB2}

case 0 0 **:** 0

case 0 1 : J2

case 1 – **:** INDB + 1

dove la scrittura *when* $\beta_{INDB} = 1$ sta a significare che il valore dell'espressione, che segue la parola chiave *do*, rappresenta lo stato successivo solo a condizione che venga effettuata la scrittura nel registro alla fine del ciclo di clock.

c) Parte Controllo

Codifichiamo, nella maniera più naturale, gli stati interni della PC (corrispondenti alle microistruzioni del microprogramma) mediante due *variabili dello stato interno*, y_0 e y_1:

stato 0 ≡ $y_0\, y_1 = 0\ 0$

stato 1 ≡ $y_0\, y_1 = 0\ 1$

stato 2 ≡ $y_0\, y_1 = 1\ -$

Dato il numero limitato di espressioni logiche da ricavare, conviene lavorare direttamente sulla struttura del microprogramma invece di riempire la tabella di verità delle due funzioni della PC si tratta di rappresentazioni del tutto equivalenti). Indicando con Y_0 e Y_1 le variabili dello stato successivo, la parte di microprogramma che ci interessa è:

$y_0y_1 = 00$ $\qquad$ (RDY0, OP_0, $OP_1 = 0 - -$) …, **$Y_0Y_1 = 00$**

(= 1 0 0) $\beta_{RDY0} = 1$, …, $\mathbf{Y_0Y_1 = 00}$

(= 1 0 1) $\beta_{RDY0} = 1$,…., $\mathbf{Y_0Y_1 = 01}$

(= 1 1 –) $\beta_{RDY0} = 1$, …, $\mathbf{Y_0Y_1 = 1–}$

y_0y_1 = 01 (I_0 = 0) …., $\mathbf{Y_0Y_1 = 01}$

(= 1) ….., $\mathbf{Y_0Y_1 = 00}$

y_0y_1 = 1– (I_0, zero (A[INDA] – B[INDB]), ACK1 = 0 0 –) …., $\mathbf{Y_0Y_1 = 1–}$

(= 0 1 –) …., $\mathbf{Y_0Y_1 = 1–}$

(= 1 – 0) …., $\mathbf{Y_0Y_1 = 1–}$

(= 1 – 1) … , $\mathbf{Y_0Y_1 = 00}$

Funzione di transizione dello stato interno di PC

La funzione di transizione dello stato interno della PC è quindi data dalle seguenti espressioni logiche :

$$Y_0 = \overline{y_0}\ \overline{y_1}\ RDY_0\ OP_0 + y_0\ \overline{I_0} + y_0\ I_0\ \overline{ACK1}$$

$$Y_1 = \overline{y_0}\ \overline{y_1}\ RDY_0\ \overline{OP_0}\ OP_1 + \overline{y_0}\ y_1\ \overline{I_0}$$

Per ogni variabile dello stato interno successivo, sono stati espressi i termini AND per i quali tale variabile vale uno, e tutti i termini AND trovati sono combinati in OR, applicando semplici riduzioni.

Funzione delle uscite della PC

La funzione delle uscite della PC, relativamente alla sola variabile di controllo β_{RDY0}, è data dalla seguente espressione logica:

$$\beta_{RDY0} = \overline{y_0}\ \overline{y_1}\ RDY_0$$

Sono stati espressi i termini AND per i quali la variabile β_{RDY0} vale uno, e tutti i termini AND trovati sono combinati in OR, applicando semplici riduzioni che, in questo caso, portano il numero di termini AND ad uno solo.

d) Prestazioni

Il *massimo ritardo della funzione delle uscite della PO* è quello per la stabilizzazione della variabile di condizionamento $x_4 = zero(A[INDA] - B[INDB])$:

$$T_{\omega PO} = t_a + t_{ALU}$$

t_a è il tempo di accesso delle memorie A e B (ovviamente, le due letture avvengono in parallelo). Ogni memoria è realizzata mediante 32 memorie da 2K poste tra loro in parallelo; ciascuna memoria da 2K viene quindi indirizzata dagli 11 bit meno significativi di IND (INDA o INDB). Le uscite delle memorie da 2K sono ingressi di un commutatore K_{mem}, le cui 5 variabili di controllo sono i corrispondenti bit più significativi di IND. Ogni porta AND di K_{mem} ha dunque 6 ingressi, e quindi è realizzabile con un solo livello di logica. Le 32 uscite delle porte AND vanno invece combinate in OR mediante un albero a 2 livelli. Dunque, essendo il tempo di accesso t_{a0} delle memorie di 2K uguale a $5t_p$:

$$t_a = t_{a0} + t_{Kmem} = 5t_p + 3t_p = 8t_p \quad , \quad T_{\omega PO} = 8t_p + 5t_p = 13t_p$$

Per ricavare *il massimo ritardo della funzione di transizione dello stato interno della PO*, occorre considerare i ritardi di stabilizzazione di tutte le operazioni che compaiono nelle microoperazioni. Non considerando i semplici trasferimenti tra registri e operazioni sugli indicatori di interfaccia, tutte con ritardo nullo, si ha:

$X \rightarrow A[J1]$: è il tempo di stabilizzazione del selezionatore di ingresso della memoria A. Trattandosi di una rete combinatoria ad un solo livello di logica, in base a quanto prima discusso si ha un ritardo:

$t_{a0} + t_p = 6t_p$.

In assenza di altre informazioni, il ritardo di scrittura per una memoria data si assume uguale al ritardo di lettura (anche se, presumibilmente, sarà inferiore).

$A[INDA] \rightarrow B[INDB]$: durante il ritardo di stabilizzazione della lettura di A, in parallelo avviene anche la stabilizzazione del selezionatore in ingresso a B (si ricordi che il selezionatore opera sul β di abilitazione alla scrittura, non sul dato d'ingresso). Dunque, il ritardo è

$t_a = 8t_p$.

$INDA + 1 \rightarrow INDA$, e tutti i casi analoghi di incremento/decremento di un registro. Poichè un commutatore a due o tre ingressi è realizzabile a due livelli di logica, il ritardo è dato da

$t_{ALU} + t_K = 5t_p + 2t_p = 7t_p$

Dunque:

$T_{\sigma PO} = 8t_p$

Il *massimo ritardo della funzione di transizione dello stato interno della PC* si ricava dalle espressioni di Y_0 e Y_1 ottenute in precedenza:

$T_{\sigma PC} = 2t_p$

Il *massimo ritardo della funzione delle uscite della PC* si deduce rapidamente osservando che:

i) il massimo numero (upper bound) di variabili presenti in un termine AND di una qualunque variabile di controllo è cinque (due variabili dello stato interno e al massimo tre variabili di condizionamento, come avviene nelle microistruzioni 1 e 3);

ii) il massimo numero di termini AND combinati in OR, per una qualunque variabile di controllo, è cinque, come si ricava osservando che la variabile con un tale numero di "1" nella tabella di verità è β_I (nel microprogramma compaiono cinque microoperazioni che provocano la scrittura nel registro I).

Dunque, anche la funzione delle uscite della PC è realizzabile a due livelli di logica:

$T_{\omega PC} = 2t_p$

Il ciclo di clock è quindi dato da:

$\tau = T_{\omega PO} + max\,(T_{\omega PC} + T_{\sigma PO}\,,\, T_{\sigma PC}) + \delta = 13t_p + 2t_p + 8t_p + t_p = 24t_p$

L'operazione esterna 2 viene eseguita come segue: una volta viene eseguita la microistruzione 0 (inizializzazione), 256 volte la microistruzione 1 (loop), ed una volta ancora la microistruzione 1 (uscita dal loop); dunque il suo tempo medio di elaborazione è dato da:

$T = 258\,\tau = 6192\,t_p$

Domanda 2

a) *ADD Ri, Rj, Rk*

Campi del formato su singola parola:

- codice operativo per "ADD" con i valori di entrambi gli operandi contenuti in registri generali (8 bit),
- indirizzo *i* di registro generale RG (6),
- indirizzo *j* di registro generale RG (6),
- indirizzo *k* di registro generale RG (6),
- inutilizzati (6).

Semantica: il valore della somma dei contenuti dei registri generali di indirizzo *i* e *j* viene scritto nel registro generale di indirizzo *k*; il contatore istruzioni viene incrementato di 1:

$$RG[i] + RG[j] \rightarrow RG[k],\ IC + 1 \rightarrow IC.$$

Sono possibili varianti in cui il primo o il secondo operando è una costante esprimibile con 6 bit (codice operativo diverso dal precedente).

b) *LOAD Ri, Rj, Rk*

Campi del formato su singola parola:

codice operativo per "LOAD" con i valori di entrambi gli addendi (base e indice), per il calcolo dell'indirizzo di memoria, contenuti in registri generali (8 bit),

- indirizzo *i* di registro generale RG (6),
- indirizzo *j* di registro generale RG (6),
- indirizzo *k* di registro generale RG (6),
- inutilizzati (6).

Semantica: il contenuto della locazione di memoria, il cui indirizzo è dato dalla somma dei contenuti dei registri generali di indirizzo *i* e *j*, viene scritto nel registro generale di indirizzo *k*; il contatore istruzioni viene incrementato di 1:

$$MEM[\ RG[i] + RG[j]\ ] \rightarrow RG[k],\ IC + 1 \rightarrow IC.$$

Sono possibili varianti in cui il primo o il secondo addendo per il calcolo dell'indirizzo è una costante esprimibile con 6 bit (codice operativo diverso dal precedente).

c) *IF pred Ri, Rj, offset*

Campi del formato su singola parola:

- codice operativo per "IF pred" con *pred* uno dei predicati diadici ammessi (8 bit),
- indirizzo *i* di registro generale RG (6),
- indirizzo *j* di registro generale RG (6),
- costante con segno (12).

Semantica: se i contenuti dei registri generali di indirizzo *i* e *j* soddisfano il predicato *pred*, allora al contatore istruzioni viene sommata la costante con segno *offset*, altrimenti il contatore istruzioni viene incrementato di 1:

$$\textbf{\textit{if}}\ pred\ (\ RG[i], RG[j]\ )\ \textbf{\textit{then}}\ IC + offset \rightarrow IC\ \textbf{\textit{else}}\ IC + 1 \rightarrow IC.$$

d) *GOTO Ri*

Campi del formato su singola parola:

- codice operativo per “GOTO” con indirizzo di salto indiretto via registro (8 bit),
- indirizzo *i* di registro generale RG (6),
- inutilizzati (18)

Semantica: il contenuto del registro generale di indirizzo *i* viene scritto nel contatore istruzioni:

$RG[i] \rightarrow IC$.