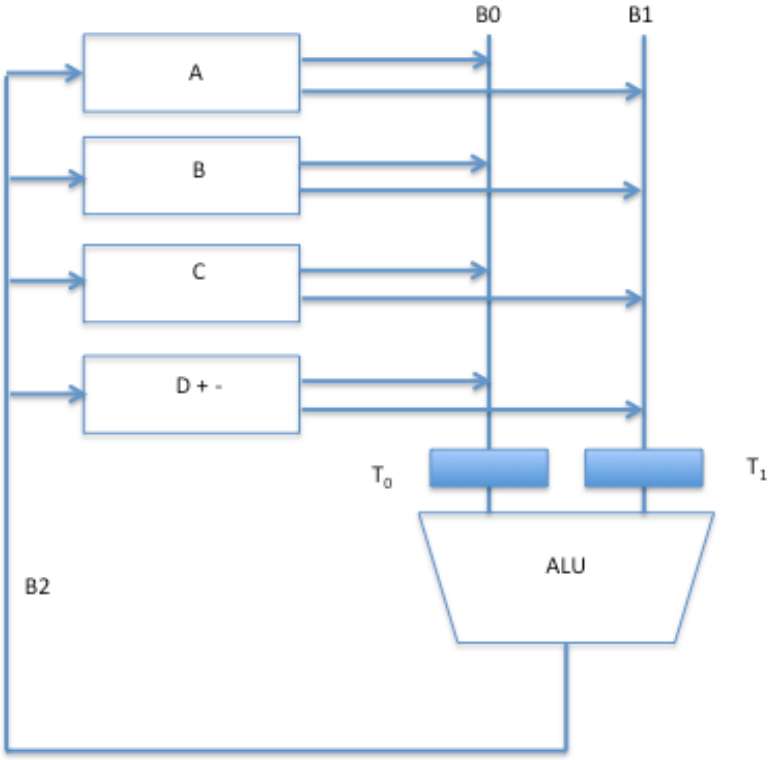
PROGETTAZIONE UNITA' DI CONTROLLO

Si consideri un sistema di elaborazione la cui Parte Operativa è schematizzata in figura. Essa contiene:

- quattro registri A, B, C e D collegati in lettura ai bus B0 e B1, ed in scrittura al bus B2 sul quale viaggia l'output dell'ALU. Il registro D è un registro contatore a incremento e decremento
- due registri tampone per stabilizzare gli input dell'ALU
- una ALU che esegue 4 operazioni: (1) la somma (output(alu) =T0+T1), (2) l'OR (output(alu) =T0 or T1), (3) la sottrazione (output(alu)=T0-T1)) e (4) la funzione identità (output (alu)=T1).



Parte Operativa

Il sistema di elaborazione deve eseguire le seguenti istruzioni:

- I_0: A or B → A;
- I_1: A*B → A, con B>0;
- I2: A div B → D, con A ≥ 0, B>0.

Si vuole progettare l'Unità di Controllo di tale sistema di elaborazione. Essa riceve in ingresso i segnali che indicano quale delle possibili istruzioni la Parte Operativa deve eseguire, e produce i segnali di controllo da inviare a quest'ultima. A tal fine:

1. Generare, per ognuna delle tre istruzioni, i relativi microprogrammi
2. Definire i segnali di controllo della Parte Operativa
3. Codificare i microprogrammi assegnando opportuni valori ai segnali di controllo
4. Completare l'Unità di Controllo con eventuali componenti necessari per supportare i segnali di condizione
5. Progettare la rete sequenziale che implementa l'Unità di Controllo

MICROPROGRAMMI

Un microporgramma è una sequenza di microistruzioni. Una microistruzione è una istruzione atomica direttamente eseguibile dai componeneti della Parte Operativa. Si noti che l'ALU è in grado di eseguire l'OR, ma non le moltiplicazioni e le divisioni, per implementare le quali sarà quindi necessario ricorrere, rispettivamente, alle operazioni di somma e sottrazione.

I0: A or B→ A

// Si assuma che i registri A e B siano stati già caricati con i valori da mettere in OR

a. A → T_0; B → T_1;
b. Alu(T_0 OR T_1) → A
c. HALT

I1: A * B → A

// esegui **A + A→ A** B-1 volte**;**
// Si assuma che i registri A e B siano stati già caricati con i valori da moltiplicare

d. B → T_1;
e. Alu(T_1)→ D;
f. Decr(D); A → T_1; //il valore in D è pari a B-1
g. if D = = 0 Φ goto j; // Φ è l'istruzione vuota
h. A→ T_0;
i. Alu(T_0 +T_1) → A; Decr(D); goto g;
j. HALT

I2: A div B → D; A mod B → A

//esegui **A-B → A** fino a quando si verifica la condizione A<B
// Si assuma che i registri A e B siano stati già caricati con i valori da dividere

k. Azzera D; A → T_0; B → T_1;
l. if T_0 < T_1 then Φ goto o
m. Alu(T_0-T_1) → A; Incr(D);
n. A → T_0 goto l
o. HALT

VARIABILI DI CONTROLLO

ALU

L0	L1	Operazione
0	0	A+B
0	1	A-B
1	0	A or B
1	1	B

Registri

- **Segnali di abilitazione alla scrittura: A_A, A_B, A_C, A_D, A_{T0}, A_{T1}**
- **Segnali di abilitazione alla lettura:**
  - Bus B0: S_A, S_B, S_C, S_D
  - Bus B1: R_A, R_B, R_C, R_D

Registro Contatore D

A_D	K	G	Z (azzeramento)	operazione
0	-	-	-	Nulla

1	**0**	**-**	**0**	**Carica dal bus B_2**
1	**0**	**-**	**1**	**Azzeramento**
1	**1**	**0**	**-**	**Incremento**
1	**1**	**1**	**-**	**Decremento**

CODIFICA MICROISTRUZIONI (alcuni esempi)

Ogni microistruzione che appare in un microprogramma viene codificata assegnando ai segnali di controllo opportuni valori (di seguito si riportano solo alcuni esempi).

microIstruzione	**A_A**	**A_B**	**A_C**	**A_D**	**A_{T0}**	**A_{T1}**	S_A	S_B	S_C	S_D	R_A	R_B	R_C	R_D	**K**	**G**	**L0**	**L1**	**Z**
A→ T_0	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**-**	**-**	**-**	**-**	**-**
Alu(T_0+T_1)→A	**1**	**0**	**0**	**0**	**0**	**0**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**0**	**0**	**-**
Incr(D)	**0**	**0**	**0**	**1**	**0**	**0**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**1**	**0**	**-**	**-**	**-**
Φ	**0**	**0**	**0**	**0**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**

L'istruzione vuota **Φ** è una istruzione che non modifica il contenuto dei registri che contengono i dati. Essa pertanto si codifica mettendo a 0 i segnali di abilitazione A_x, dove x ∈ {A, B, C, D}.

SEGNALI DI CONDIZIONE

Le due istruzioni condizionali nei microprogrammi delle istruzioni I1 e I2

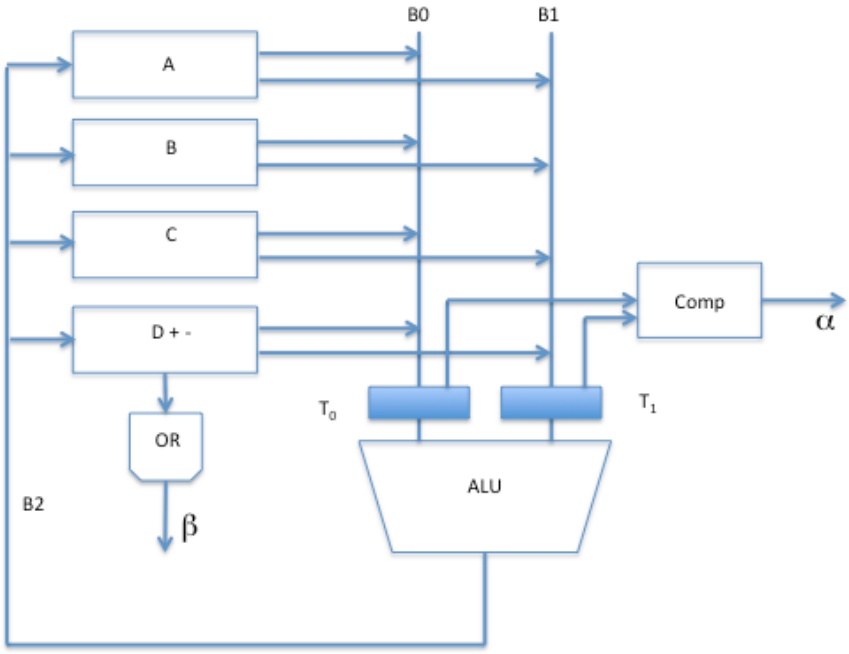
- if D = = 0 **Φ** goto j;
- if $T_0 < T_1$ then **Φ** goto m

richiedono circuiti aggiuntivi per il controllo delle condizioni. In particolare:

- la condizione D==0 viene testata aggiungendo una porta OR che riceve in ingresso i bit $d_1, \dots d_n$ del registro D; ovviamente D == 0 sse $\alpha = OR(d_1, \dots d_n) = 0$;

- la condizione $T_0 < T_1$ richiede l'aggiunta di un circuito comparatore che riceve in ingresso i bit dei registri T_0 e T_1 e fornisce in output un segnale $\beta = 1$ sse $T_0 < T_1$.

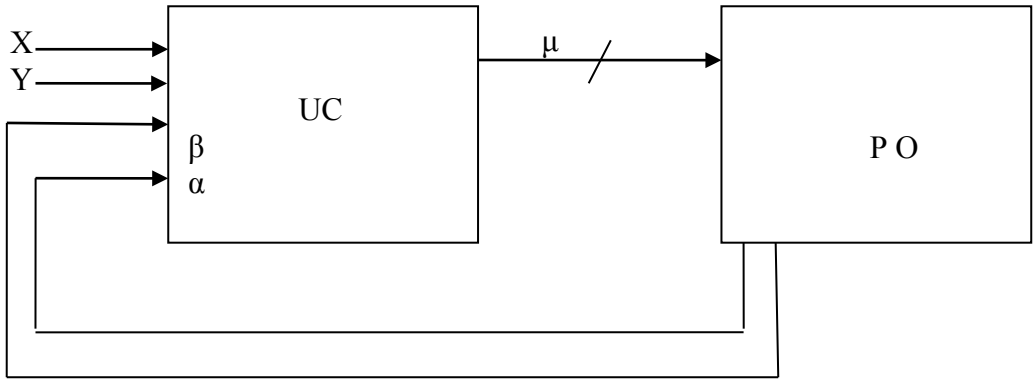
Pertanto, la Parte Operativa genera due segnali di condizione α e β che vengono inviati all'Unità di Controllo.



SCHEMA ARCHITETTURALE DEL SISTEMA DI ELABORAZIONE

Il sistema di elaborazione consiste di una Parte Operativa ed una Unità di Controllo. L'Unità di Controllo

- riceve in ingresso i segnali X,Y che indicano l'istruzione da eseguire (00 esegue la I_0, 01 esegue la I_1, 10 esegue la I3, 11 esegue l'istruzione vuota), più i segnali di condizione α e β inviati dalla Parte Operativa
- fornisce in uscita i 19 segnali di controllo μ che codificano le microistruzioni.

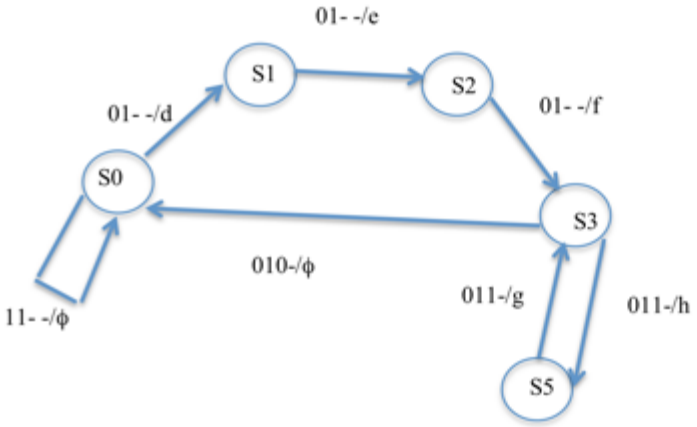


PROGETTAZIONE UNITA' DI CONTROLLO (AUTOMA A STATI FINITI)

L'Unità di Controllo è un automa a stati finiti. Infatti, il suo comportamente può essere descritto attraverso un grafo delle transizioni che si ottiene come segue:

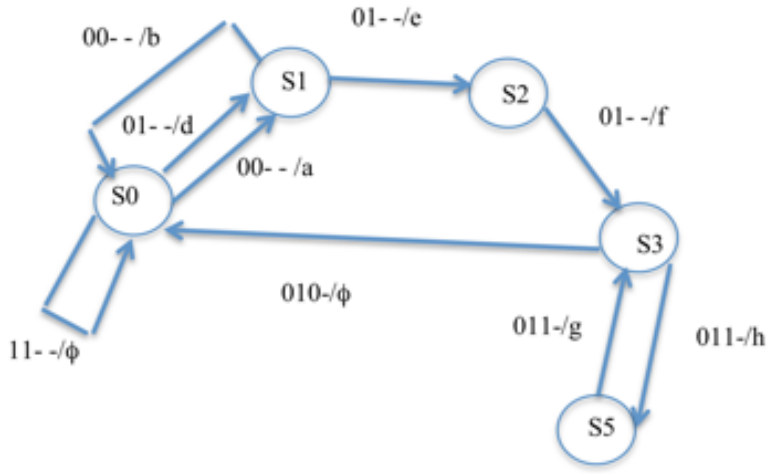
- si parte dal microprogramma più lungo (quello associato alla I_1); S_0 è lo stato iniziale nel quale il sistema permane fintantoché X=1 e Y=1; quando X=0 e Y=1 (istruzione I_1) il sistema evolve nello stato S_1 (indipendentemente dai valori di α e β) inviando alla PO i segnali di controllo che codificano la microistruzione *d* (prima microistruzione del microprogramma di I_1); poi passa nello stato S_2, eseguendo la microistruzione *e*, ecc. ecc. Si noti che quando l'automa si trova nello stato S_3,

lo stato successivo dipende dal valore del segnale di condizione α; in particolare, se α=0 (cioè, D=0) allora l'automa evolve nello stato S_0 (fine del microporgramma), altrimenti evolve nello stato S_4 (gli stati S_3, S_4 e S_5 rappresentano il ciclo racchiuso tra le microistruzioni *g* ed *i*).



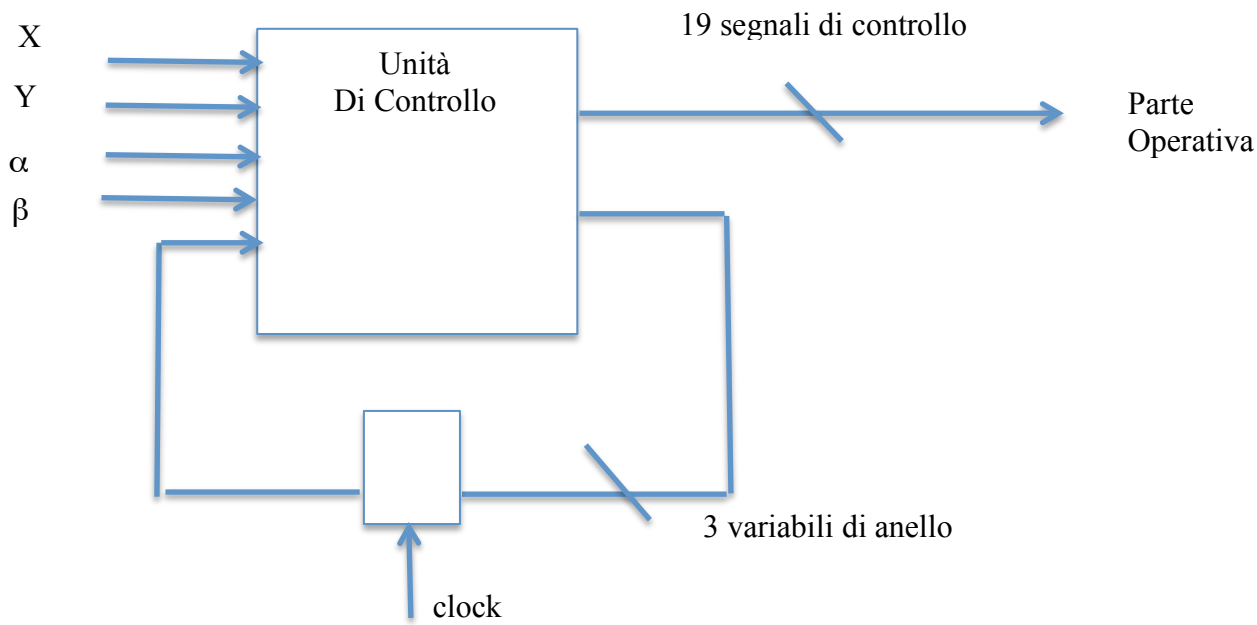
Grafo delle transizioni per il microprogramma I1

- Utilizzando un sottoinsieme dei suddetti stati S_0-S_4, si rappresentano anche i microprogrammi delle istruzioni I_0 e I_2 (il grafo delle transizioni per I_0, unitamente a quello per I_1, è riportato nella seguente figura – si lascia per esercizio la rappresentazione del grafo per I_2).



Grafo delle transizioni per i microprogrammi I0 e I1

L'UC è quindi una macchina sequenziale (automa a stati finiti) sincrona con 4 segnali di ingresso (X, Y, α e β), 3 variabili di anello (necessarie per rappresentare i 5 stati) e 19 segnali di uscita (segnali di controllo della Parte Operativa).



La parte combinatoria dell'UC è una rete con 7 variabili d'ingresso (X, Y, α e β, più le 3 variabili di anello) e 22 uscite (i 19 segnali di controllo per la Parte Operativa più le 3 variabili di anello). Il comportamento ingresso-uscita di tale rete si evince facilmente dal grafo delle transizioni.