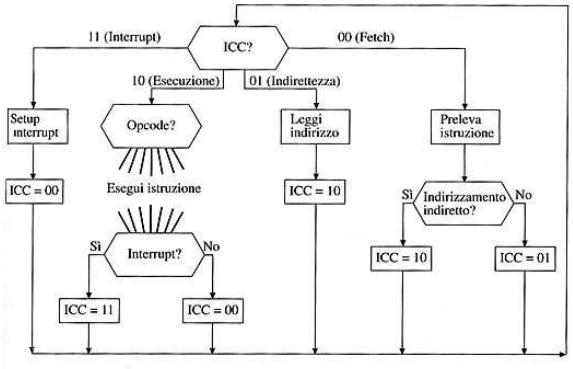
Ciclo di Istruzione

- Può essere suddiviso in 4 tipi di sequenze di micro-operazioni (cioè attività di calcolo aritmetico/logico, trasferimento e memorizzazione dei dati), non tutte necessariamente da realizzare per ogni istruzione:
  - 0: fetch (prelievo dell'istruzione)
  - 1: indirettezza (calcolo indirizzi operandi e loro recupero)
  - 2: esecuzione (varia al variare del codice operativo)
  - 3: interruzione (controllo ed eventuale attivazione di una procedura di gestione dell'interruzione)

Ciclo di Istruzione

- Le 4 tipologie di ciclo possono essere codificate tramite 2 bit
  - 0: fetch → 00
  - 1: indirettezza → 01
  - 2: esecuzione → 10
  - 3: interruzione → 11
- Si utilizza un registro di 2 bit, l'***instruction cycle code*** (ICC), per tener traccia di quale fase dell'istruzione è correntemente in atto

Ciclo di Istruzione (diagramma di flusso)



Controllo

- La Parte (o Unità) Controllo (PC) della CPU si fa carico di realizzare il flusso di controllo appropriato per ogni istruzione tramite l'invio di opportuni segnali di controllo alla Parte Operativa
- **Requisiti funzionali** (cioè le funzioni che la PC deve eseguire):
  - Definire gli elementi di base del processore
  - Definire le micro-operazioni che il processore esegue
  - Determinare le funzioni che la PC deve effettuare per l'esecuzione delle micro-operazioni

Elementi Base

- Come visto in precedenza gli elementi di base sono:
  - ALU
  - Registri
  - Bus dati interno
  - Bus dati esterno
  - Unità di controllo

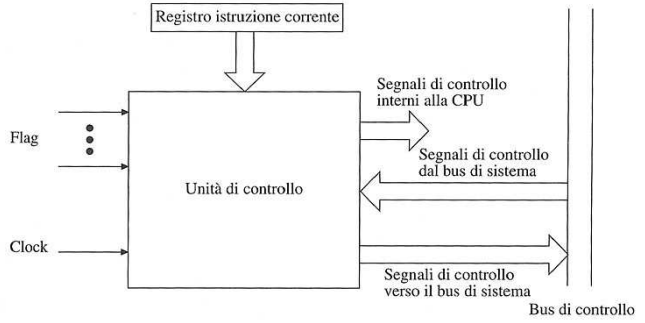
Tipologie di micro-istruzioni

- Come visto in precedenza, le micro-istruzioni ricadono tutte nelle seguenti tipologie:
  - Trasferimento dati da un registro all'altro
  - Trasferimento dati da un registro ad un'interfaccia esterna
  - Trasferimento dati da un'interfaccia esterna ad un registro
  - Esecuzione di una operazione aritmetica o logica, che utilizzi i registri come input e output

Funzioni della PC

- Quindi i compiti base della PC sono:
  - **Serializzazione**: determina la "giusta" sequenza di micro-operazioni da eseguire in funzione del codice operativo dell'istruzione
  - **Esecuzione**: provoca l'esecuzione di micro-operazioni
- La realizzazione di questi compiti base passa attraverso la generazioni di opportuni *segnali di controllo*

Segnali di Controllo



Realizzazione della PC

- Ci sono due alternative:
  - **Cablata**:
    - si realizza direttamente tramite circuiti digitali (livello di astrazione 0);
    - soluzione tipica di architetture RISC;
  - **Microprogrammata** (la trattiamo di seguito):
    - si realizza tramite microprogrammazione (livello di astrazione 1);
    - soluzione tipica di architetture CISC;
    - permette una maggior flessibilità in fase di progettazione: rende facile modificare le sequenze di micro-operazioni

Microarchitettura: Livello 1

- Sistema visto come composizione di n **unità di elaborazione interagenti ed autonome** (dati gli ingressi ottenuti tramite interazione con altre unità)
- Ad ogni unità è affidato un certo **sottinsieme** delle funzionalità del sistema
- L'interazione tra tali unità fornisce la funzione complessiva di sistema
- Un elaboratore generico comprende molte unità (CPU, memoria, I/O, etc.)

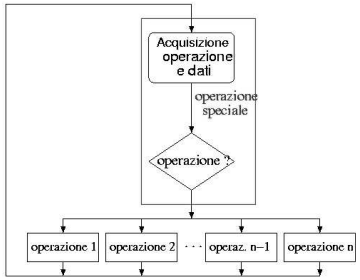
Livello 1 (segue)

- Il livello 1 è anche detto **livello firmware**
- Il livello firmware è interpretato direttamente (eseguito) dall'hardware (livello 0: circuiti digitali)
  - mediante l'interconnessione di reti combinatorie e sequenziali
- Ciascuna unità opera in modo **Sequenziale**, con funzionamento specificato da un *microprogramma (µprogramma)* espresso in un dato *micro-linguaggio (µlinguaggio)* che permette di programmare la interazione fra le reti combinatorie e sequenziali del livello 0

Livello Firmware

- Ogni unità a livello firmware è in grado di eseguire sequenze di operazioni
- Le operazioni che formano tali sequenze, dette operazioni esterne dell' unità , sono **indipendenti fra loro** ed appartengono ad un insieme predefinito
- L'esecuzione di ogni diversa operazione esterna è descritta da un diverso frammento di *µprogramma* (detto interprete dell'operazione)
- Esiste anche una operazione speciale che di fatto realizza l'acquisizione dall'esterno dei dati e del codice della operazione esterna da eseguire

Livello Firmware

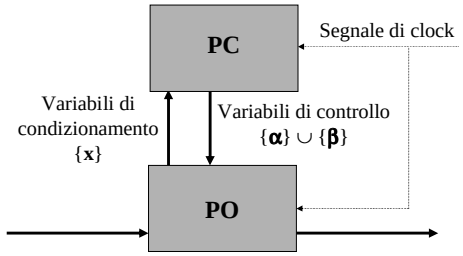


- Il *μprogramma* di una unità riunisce i frammenti di programma delle diverse operazioni (esterne e speciale)
- Il *μprogramma* ha una struttura ciclica in cui si alterna l'esecuzione della operazione speciale con l'esecuzione della operazione esterna il cui codice e dati da eleborare sono stati acquisiti dalla operazione speciale

Livello Firmware

- L'interpretazione del microprogramma viene usualmente eseguita da due reti sequenziali LLC interagenti, denominate **Parte Controllo** (PC) e **Parte Operativa** (PO)
- PC è l'automa di controllo dell'unità
  - ha tanti stati interni quante sono le μistruzioni del μprogramma (lo stato corrente determina la μistruzione in esecuzione)
- PO esegue le operazioni elementari previste da ogni μistruzione
  - quindi dispone di: registri, reti combinatorie (ALU, …), circuiti per l'instradamento dei dati (multiplexer, bus, …)

Livello Firmware (segue)



Livello Firmware (segue)

- La PO provvede all'esecuzione dei comandi del microlinguaggio (**microistruzioni**) tramite reti combinatorie standard e registri
- La PC provvede a:
  - Controllo di sequenzializzazione delle microistruzioni tramite **variabili di condizionamento** {**x**} relative allo stato interno di PO
  - Invio comandi di esecuzione a PO tramite **variabili di controllo** {**α**} ∪ {**β**}

Livello Firmware (segue)

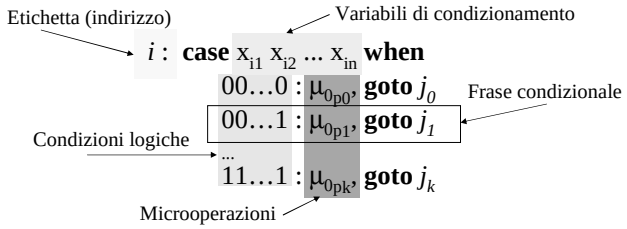
- Variabili di condizionamento {**x**} : PO → PC
  - Esempio: test per zero del contenuto di un certo registro
- Variabili di controllo : PC → PO
  - {β} abilita / disabilita la scrittura nei registri di PO
  - {α} fornisce gli ingressi secondari per i commutatori, selezionatori, ALU, etc. di PO
    - Esempio: instradamento da registro/i sorgente a registro destinazione della microistruzione

Livello Firmware (segue)

- **Progettazione di unità**
  1. Specifica delle operazioni esterne dell'unità
  2. Scrittura del microprogramma di interpretazione delle operazioni esterne
  3. Derivazione di PO a partire dal microprogramma
  4. Derivazione di PC a partire dal microprogramma
  5. Determinazione del periodo di clock
  6. Valutazione delle prestazioni

Livello Firmware (segue)

- Ogni microistruzione del microprogramma esegue in un ciclo di clock
- Microlinguaggio Phrase Structured (**PS**)
- Microoperazioni nulle (*nop*) o di trasferimenti, anche multipli, tra registri

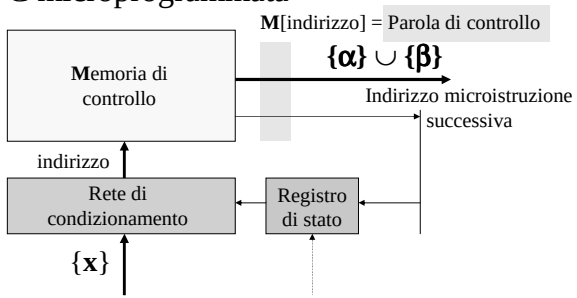


Livello Firmware (segue)

- **PO** tipicamente vista come rete sequenziale di **Moore**
  - Per microoperazioni non parallele basta una ALU multi-funzione
- **PC** come rete sequenziale di **Mealy**
  - Etichette microistruzioni ⇔ stati interni
  - Variabili di condizionamento ⇔ stati ingresso
  - Microoperazione (=variabili di controllo) ⇔ stati di uscita

Livello Firmware (segue)

- PC microprogrammata



- **{α} ∪ {β}** designa la microoperazione richiesta

Modelli PC-PO alternativi

- **Moore-Moore**, microlinguaggio Transfer-Structured (**TS**)
  - Maggior numero di cicli di clock ma di minor durata rispetto al modello Mealy-Moore
- **Moore-Mealy**
  - Equivalente al modello Moore-Moore, con PO ottenuta *anticipando* il prelievo di {x} all'*ingresso* dei registri

Livello Firmware (segue)

- Microlinguaggio TS

```
etichetta : microoperazione
        case condizione logica
                when valore 1 => indirizzo successivo 1
                when valore 2 => indirizzo successivo 2
                …
                when valore n => indirizzo successivo n
        end case
```


Esempio

Specifica:
while true **loop**
A := A + B;
if A < 0 **then** A :=- A**end if**;
end loop;

Formalismi di μprogramma:
; : azioni *alternative*
, : azioni *parallele*
Le condizioni logiche si *omettono* se *tutte* abilitano la *stessa* azione

PS (Mealy-Moore): **[*ottimizzato*]**
0. A + B → A, 1
1. (A_0 = 1) –A → A, 0 ; (A_0 = 0) A + B → A, 1

TS (Moore-Moore):
0. A + B → A, 1
1. *nop* (A_0 = 0) 0 ; (A_0 = 1) 2
2.- A→ A, 0

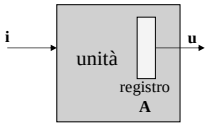
TS (Moore-Mealy):
0. A + B → A (A_0 = 0) 0 ; (A_0 = 1) 1
1.- A→ A, 0

[{x} *anticipato*]

Esempio di PC-PO

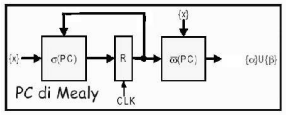
- Assunzioni:
  - una sola operazione esterna (vedi specifica sotto) che richiede un ingresso **i** (numero rappresentato in complemento a 2)
  - per semplificare non si considera l'operazione speciale, cioè si assume che:
    - l'unità riceva l'ingresso **i** quando l'unità è pronta per l'elaborazione
    - l'uscita **u** della unità (memorizzata internamente in un registro **A**) è prelevata da chi ne ha bisogno prima che questa venga sovrascritta dalla uscita successiva

Specifica:
while true **loop**
A := A + i; /* i ingresso */
if A < 0 **then** A :=- A **end if**;
end loop;

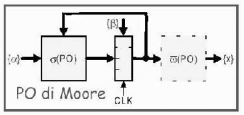


PS (Mealy-Moore): [***non ottimizzato***]
0. A + i → A, 1
1. (A_0 = 1) –A → A, 0 ; (A_0 = 0) nop, 0

PC-PO: Mealy-Moore

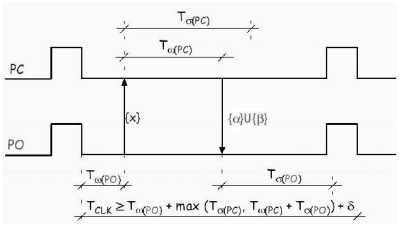


$\bar{\omega}$(PO) può mancare o essere molto semplice



I segnali di condizione derivano dalle uscite dei registri della PO

- Funzionamento seriale



- T_{CLK} è il periodo di clock
- $T_{\sigma(PC)}$: ritardo massimo della rete combinatoria σ(PC)
- $T_{\sigma(PO)}$: ritardo massimo della rete combinatoria σ(PO)
- $T_{\omega(PC)}$: ritardo massimo della rete combinatoria ω(PC)
- $T_{\omega(PO)}$: ritardo massimo della rete combinatoria ω(PO)
- δ ritardo prop.+ setup flip-flop

Esempio: PO

- La PO ha bisogno di:
  - una ALU in grado di effettuare sia l'addizione (α=0) che la negazione (per il cambio di segno; α=1)
    - in ingresso la ALU ha **i** (ingresso alla unità) e il contenuto del registro A, ed α è il segnale di controllo che seleziona l'operazione da eseguire
  - un registro A, la cui scrittura è abilitata con segnale β=1 e disabilitata con segnale β=0
    - il segno del valore contenuto in A è stabilito dal bit A_0 del registro A (A_0 costituisce il segnale di condizionamento **x**)
- lo stato interno di PO è costituito dal contenuto del registro A (se il registro è a n bit, allora la PO avrà 2^n stati interni)
- la funzione di uscita di PO (modello Moore) è l'identita (infatti l'uscita **u** coincide con il contenuto del registro A, cioè lo stato)

Esempio: PC

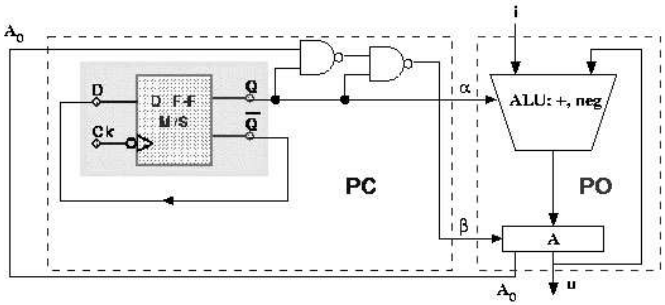
- La PC è una rete sequenziale di Mealy dotata di 1 ingresso binario (A_0), 2 uscite binarie (α, β) e 2 stati interni ("0", "1")
  - nello stato "0", qualunque sia il valore assunto da A_0, la PC deve generare i segnali α=0 (A+**i** → A) e β=1 (registro A abilitato)
  - nello stato "1", se A_0=1, la PC deve generare i segnali α =1 (A→ A) e β=1 (registro A abilitato)
  - nello stato "1", se A_0=0, la PC deve generare i segnali α =* (non importa il valore di α) e β=0 (registro A disabilitato)

Stato \ Ingresso	A_0=0	A_0=1
"0"	"1" αβ=01	"1" αβ=01
"1"	"0" αβ=*0	"0" αβ=11

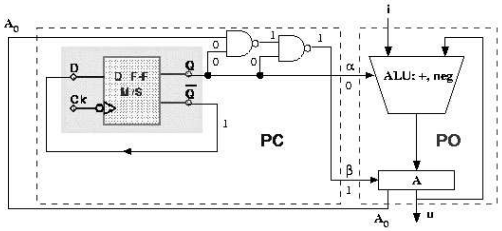
Tabella della PC

Realizzazione di PC-PO

- Solo 2 stati: uno per ogni μistruzione → si usa un flip-flop D
- Usando la tabella di PC si ottiene il seguente circuito



Esempio PC-PO

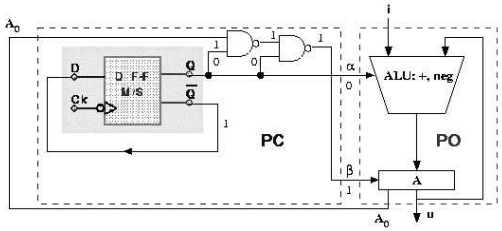


A_0=0
"0"

Stato \ Ingresso	A_0=0	A_0=1
"0"	"1" αβ=01	"1" αβ=01
"1"	"0" αβ=*0	"0" αβ=11

Tabella della PC

Esempio PC-PO

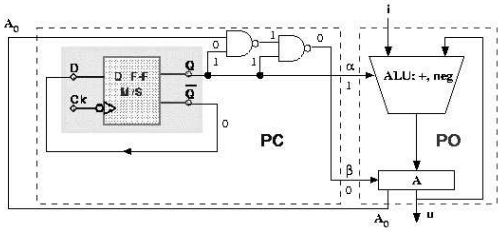


A_0=1
"0"

Stato \ Ingresso	A_0=0	A_0=1
"0"	"1" αβ=01	"1" αβ=01
"1"	"0" αβ=*0	"0" αβ=11

Tabella della PC

Esempio PC-PO

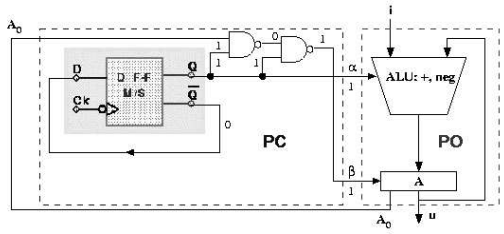


A_0=0
"1"

Stato \ Ingresso	A_0=0	A_0=1
"0"	"1" αβ=01	"1" αβ=01
"1"	"0" αβ=*0	"0" αβ=11

Tabella della PC

Esempio PC-PO



A_0=1

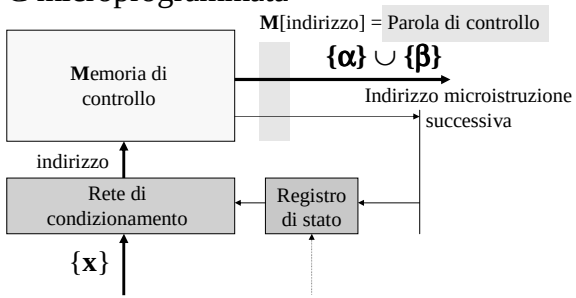
"1"

Stato \ Ingresso	A_0=0	A_0=1
"0"	"1" αβ=01	"1" αβ=01
"1"	"0" αβ=*0	"0" αβ=11

Tabella della PC

Alternativa

- PC microprogrammata



- **{α} ∪ {β}** designa la microoperazione richiesta