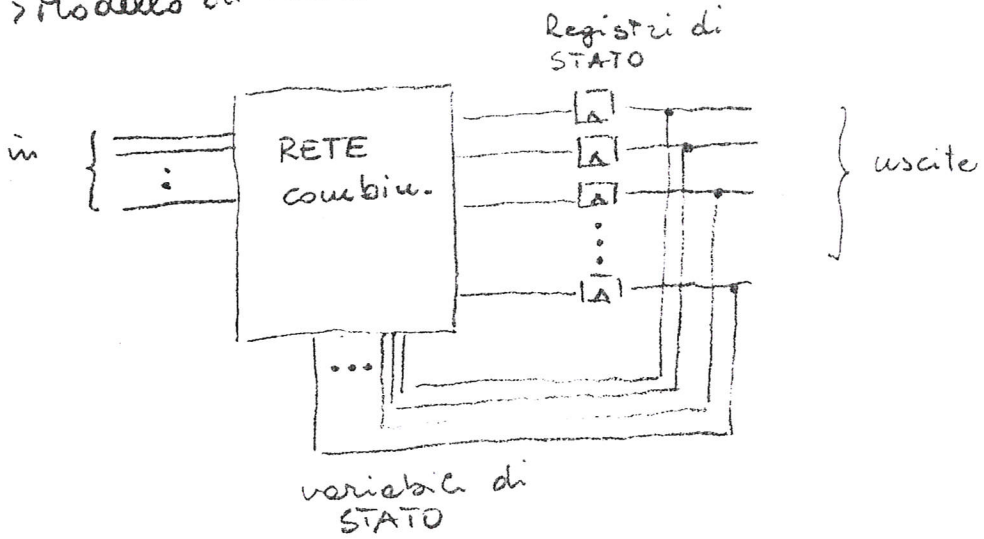
6 VERSO IL MICROPROCESSORE

- Processori: Sistemi a controllo di programma
  - Circuiti in grado di eseguire sequenze di istruzioni conservate in un dispositivo specifico (RAM, ROM)
  - Macchine sequenziali complesse sincrone
    Partiamo dal modello base di rete logica
    > Modello di Moore

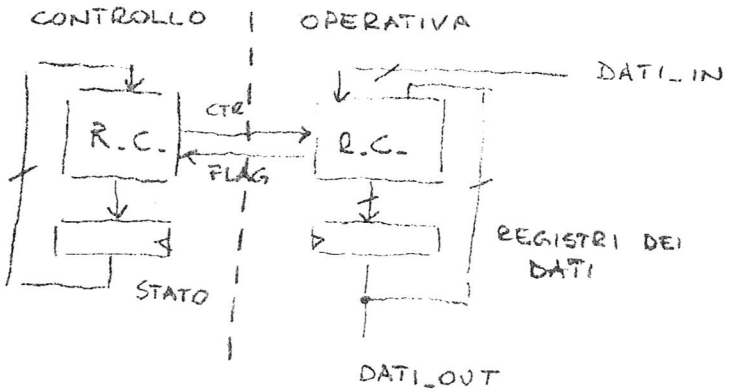


È evidente che la funzionalità della rete è contenuta nel cablaggio della rete combinatoria.

> Idea: cerchiamo di separare capacità "OPERATIVE" della capacità di gestire il flusso delle operazioni

- Alcuni registri si "specializzeranno" a supporto dell'esecuzione delle operazioni (appoggio ai dati)
- Altri serviranno per gestire il "flusso" di esecuzione del programma

- Modello di macchina divisa in PARTE OPERATIVA e PARTE CONTROLLO

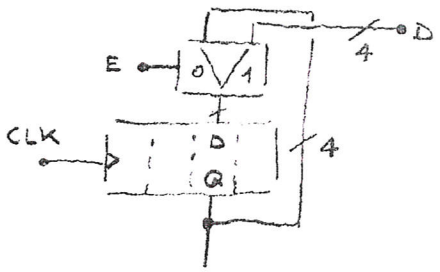


- Come può essere realizzata una PARTE OPERATIVA.

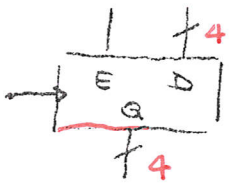
> Il progetto è fortemente influenzato dalle applicazioni per cui è pensato il processore
  - Elaborazioni di calcolo di tipo generale
  - Elaborazione di segnale
  - Rappresentazione dell'informazione, tipi di dati

> Ha una struttura "data path", formata da registri, bus, e blocchi operatori

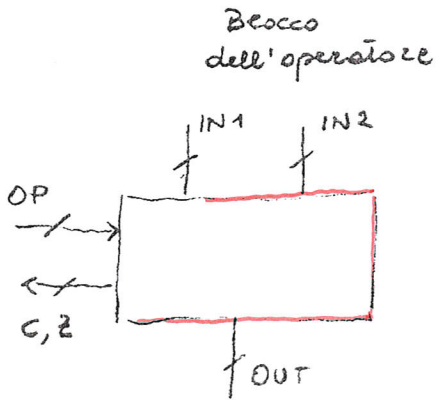
> Proponiamo, a titolo di esempio, una parte operativa con cui si possono realizzare semplici operazioni logiche e l'incremento
  - La dotiamo di 4 registri intercambiabili di uso generale (a 4 bit) con abilitazione "E"
  - Diamo la possibilità alla parte di controllo di caricare i registri con un valore "immediato"
  - Prevediamo un operatore INC, AND, NOT, XOR con operazioni logiche "bit-wise".



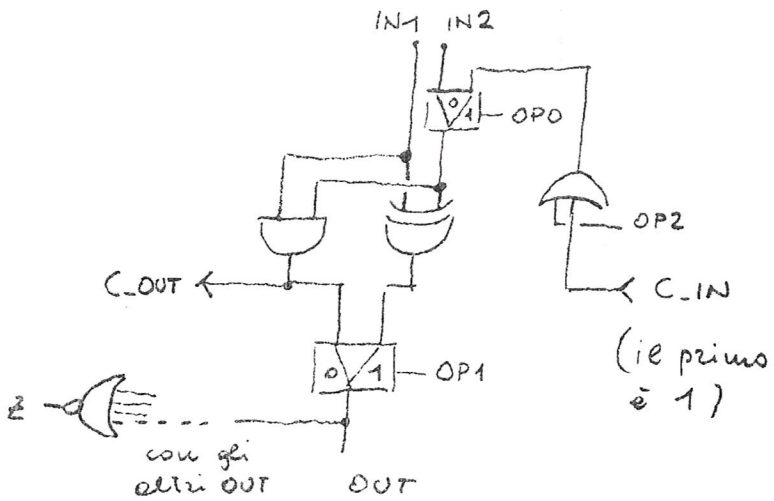
→



Registro con abilitaz.

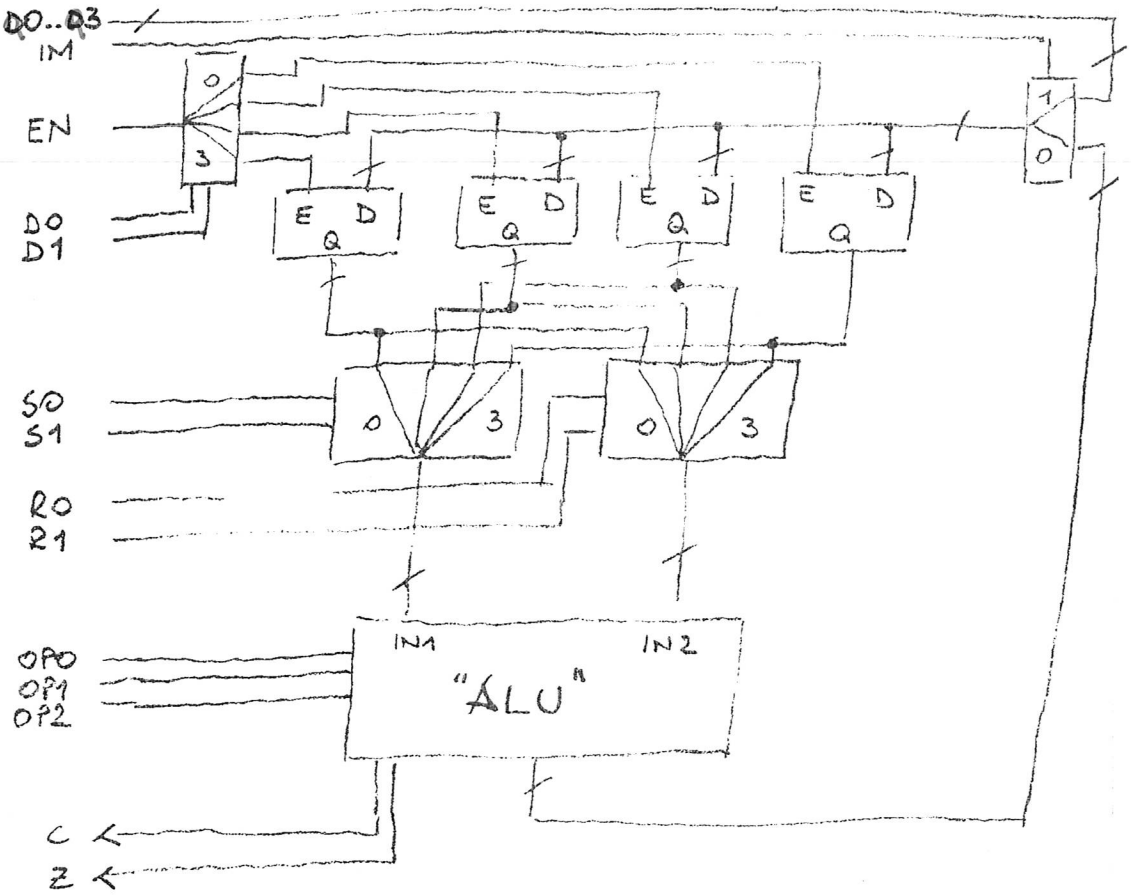


Blocco dell'operatore



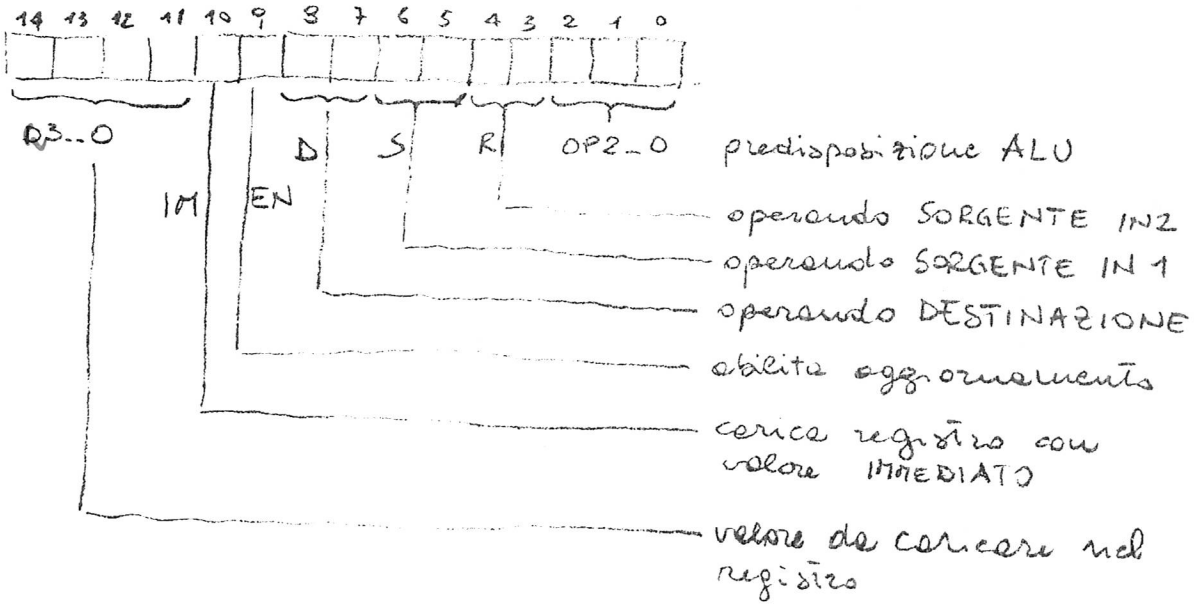
OP2	OP1	OP0	Operazione
0	0	0	AND (IN1, IN2)
0	0	1	—
0	1	1	INC (IN1)
0	1	0	XOR (IN1, IN2)
1	1	0	NOT (IN1)

• Si ha quindi:



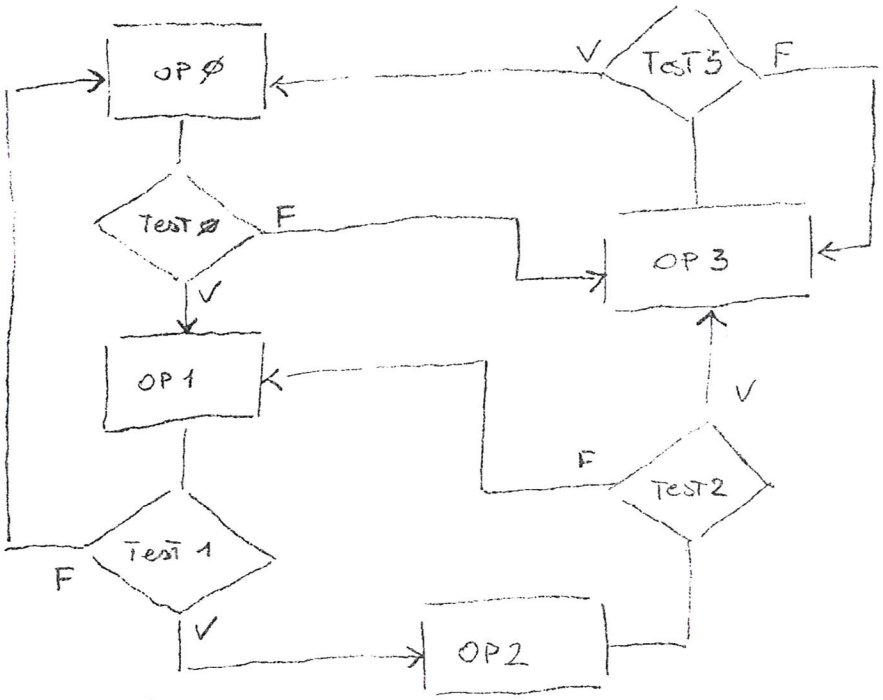
> Questa parte operativa prevede quindi "istruzioni" da 15 b, che predispongono le varie parti per eseguire quanto richiesto

• NON tutte le combinazioni dei 15 b sono significative



- Possiamo immaginare di realizzare la parte di controllo in modo "strutturato", al fine di abilitarla a eseguire flussi di operazioni MODIFICABILI sulla base delle operazioni esterne

ESEMPIO



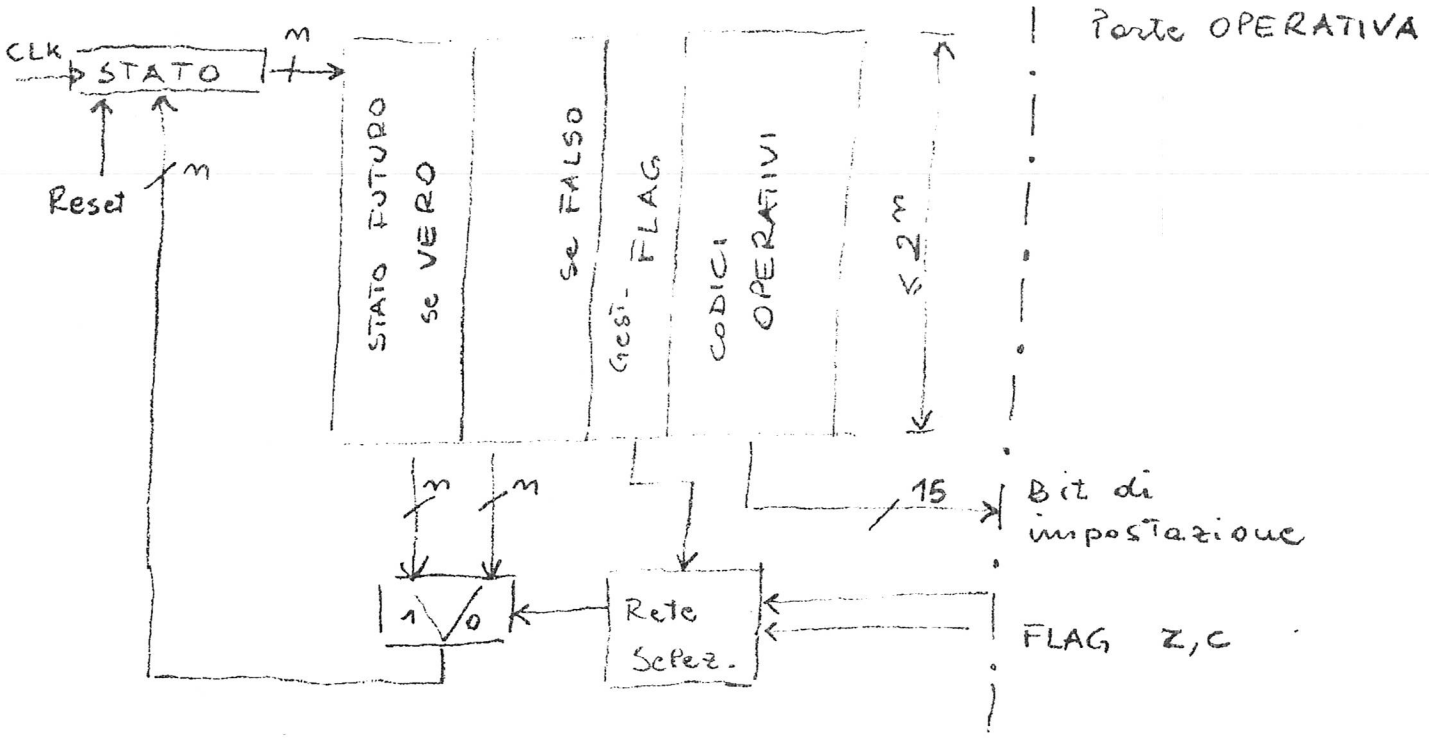
> Le OP vengono eseguite "impovendo" una certa (15 b) parola con segnali di controllo alla parte operativa

> I TEST sono eseguiti su combinazioni logiche di segnali provenienti dalla parte operativa (Z o C)

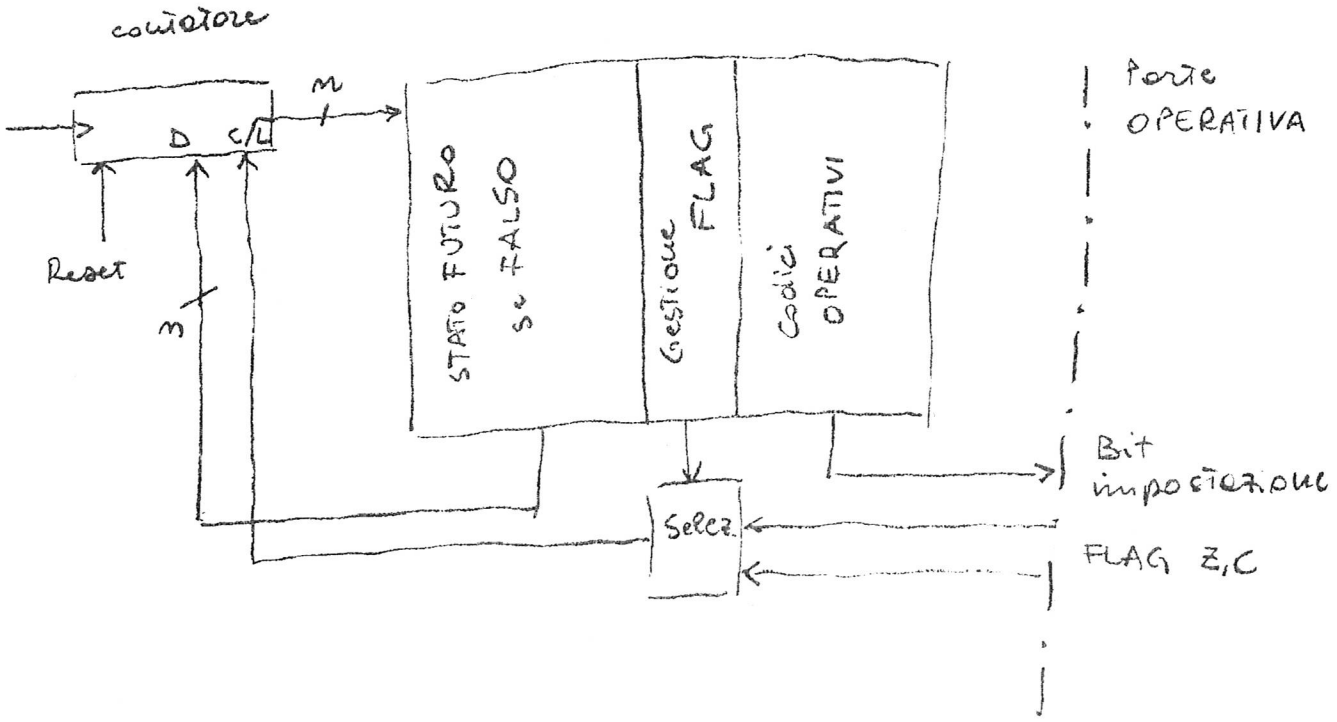
> Si usa perciò un'architettura, basata su MEMORIA, per realizzare questo diagramma di prende il nome di SEQUENZIATORE.

- Serve un REGISTRO di appoggio per indicare lo STATO corrente (n° STATI = $2^{n° bit}$)
- Serve un meccanismo per decidere lo STATO futuro, impostando i TEST
- I bit che gestiscono l'OPERAZIONE devono essere inviati alla parte operativa (sono i 15 b visti prima)

> Sequenziatore "standard"



> Se le operazioni previste nell'esecuzione del programma vanno eseguite SEQUENZIALMENTE, la struttura può essere significativamente semplificata usando un contatore con CARICAMENTO parallelo



- Possiamo vedere cosa significa programmare quella struttura in modo da realizzare, in R0, un contatore che conta continuamente da 5 a 11 e riparte ogni volta da capo.

> Si tratta di scrivere il contenuto della memoria, in modo che le operazioni siano corrette
  • Facciamo riferimento all'architettura con registro

indirizzo stato	FUTURO 1	FUTURO 0	FLAG	CODICE OP'	Comm
0	1	-	1	5, 1, 1, 1, -, -, -	R1 ← 5
1	2	-	1	11, 1, 1, 2, -, -, -	R2 ← 11
2	3	-	1	-, 0, 1, 0, 1, 1, 0	R0 ← R1
3	4	-	1	-, 0, 1, 0, 0, -, 3	INC R0, R0
4	3	2	2	-, 0, 0, -, 0, 2, 2	XOR R0, R2, -

> Siamo riusciti a ottenere una funzionalità "evoluta" solo cambiando il contenuto della memoria interna del processore, cioè il MICROCODICE.

- Limiti dell'approccio "POPC" proposto

> Le "istruzioni" sono in forma "estesa". Richiedono grandi quantità di memoria.

> la memoria del (μCODICE) influenza pesantemente le prestazioni del processore.
  • Non può essere troppo grossa
  • deve essere facile da "raggiungere" e modificare

> SOLUZIONE:
  • Usare un secondo livello per codificare le istruzioni in forma COMPATTA
  • Usare il μcodice per costruire un INTERPRETE di questo nuovo tipo di istruzioni (livello ASSEMBLER)

(CPU)

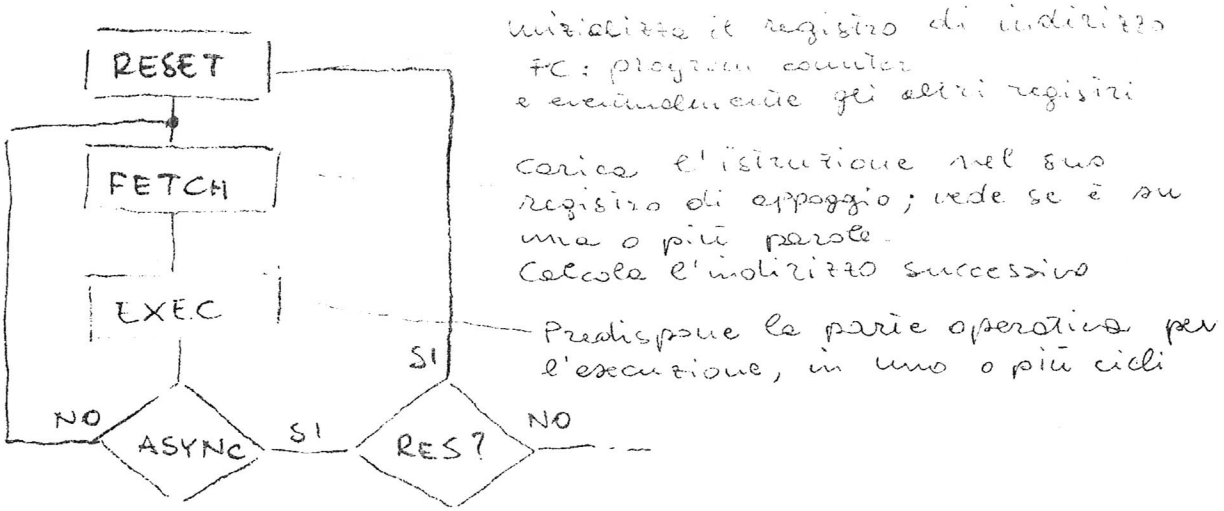
- Processore con comandi assembler

L'architettura di questo sistema prevede una parte CONTROLLO specializzata, in grado di dialogare con una memoria esterna tramite un bus

> Servono alcuni registri di appoggio, per l'indirizzo della memoria e per le istruzioni caricate da eseguire

> Occorre stabilire un flusso ciclico, che dopo una inizializzazione, prelevi sequenzialmente (riconoscendo le eccezioni) le istruzioni, le esegua.

- Si può prevedere anche la possibilità di interrompere con segnali esterni (RES, INT, HALT) questo flusso



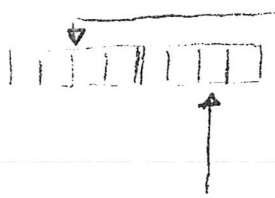
- Per "scompattare" l'istruzione assembler occorre una rete combinatoria di decodifica, che interpreti i diversi campi che sono stati inseriti nell'istruzione

> Proseguendo nel nostro esempio, possiamo immaginare un processore con il seguente assembler:

(CODICI "MNEMONICI")

```
INC  RD
AND  RD, RS
XOR  RD, RS
NOT  RD
LD   RD, RS
LDI  RD, n
JP   n
JP   c, n
NOP
```

- Possiamo organizzare i comandi su parole (1 o più) di 8 b, eseguendo vari campi



codice OPERATIVO
- indica QUALE operazione
- in CHE MODO sono indicati gli operandi e DOVE trovarli
- specifica SE il comando è su 1 o più parole
- specifica SE occorre fare salti

operandi o altre informazioni utili per eseguire l'operazione

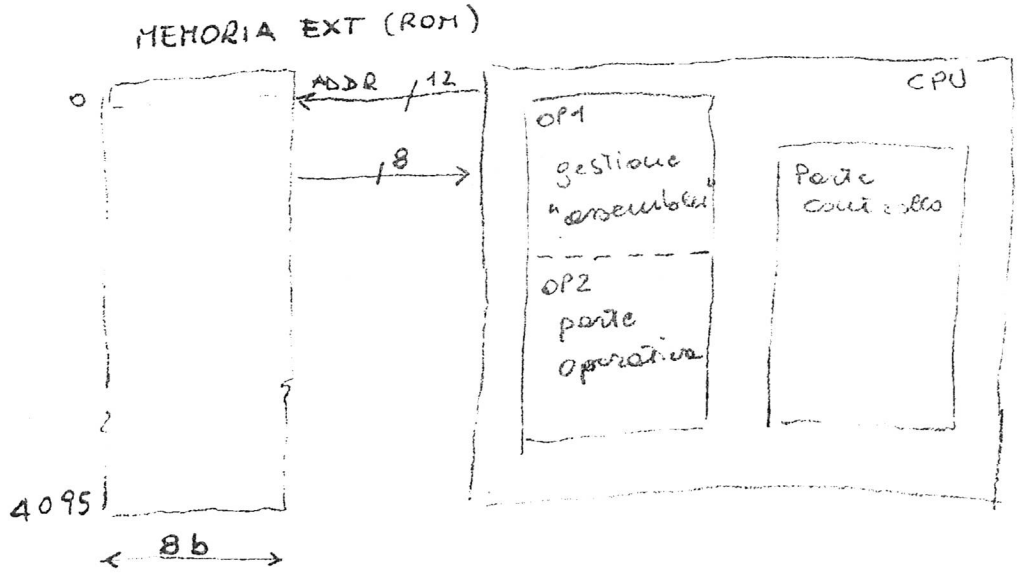
- Proviamo a costruire i codici per le istruzioni proposte (codice mnemonico → codice binario)

NOP	0 0 0 0 0 d d d	1	
LD RD,RS	0 0 0 1 D S	1	
LDI RD, n	1 0 D n	1	
NOT RD	0 0 1 0 D d0	1	
INC RD	0 0 1 1 D d0	1	
AND RD,RS	0 1 0 0 D S	1	
XOR RD,RS	0 1 0 1 D S	1	
JP c,m	1 1 c mh	ml	2
JP m	0 1 1 1 mh	ml	2

D, S	
00	R0
01	R1
10	R2
11	R3

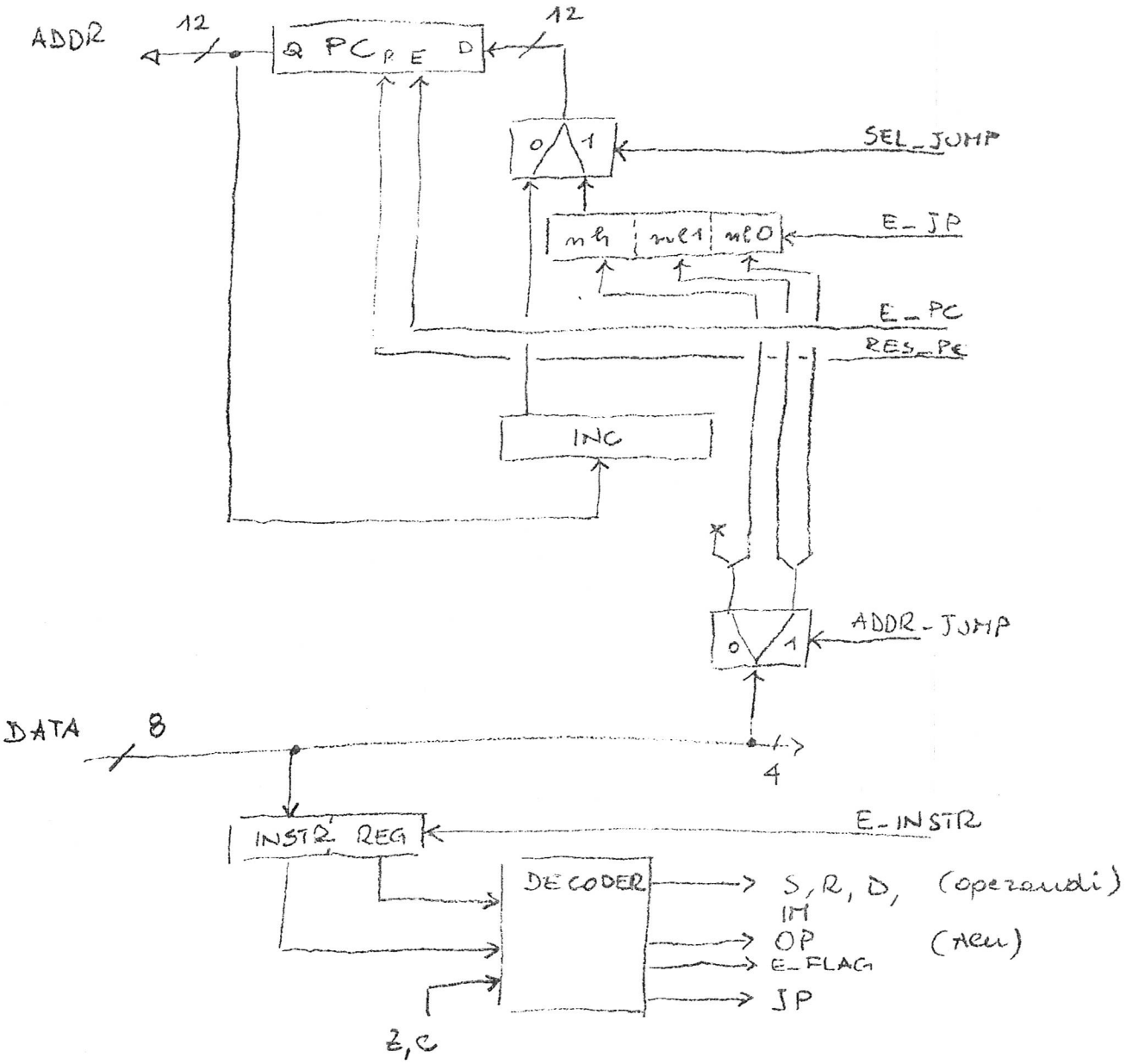
c	
00	Z
01	NZ
10	C
11	NC

- Abbiamo fissato in 4096 locazioni lo spazio di memoria e in 12 b la dimensione del registro PC

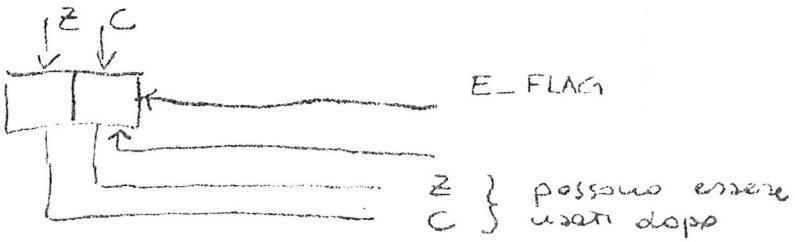


> I nuovi registri possono essere considerati alla stessa stregua di quelli della parte operativa

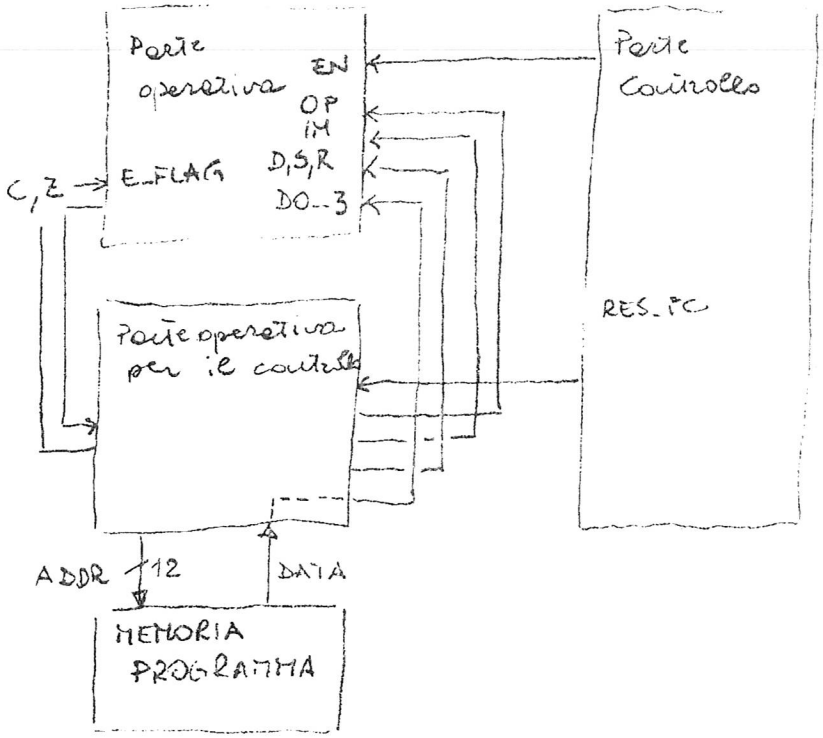
Vediamo di specificarli



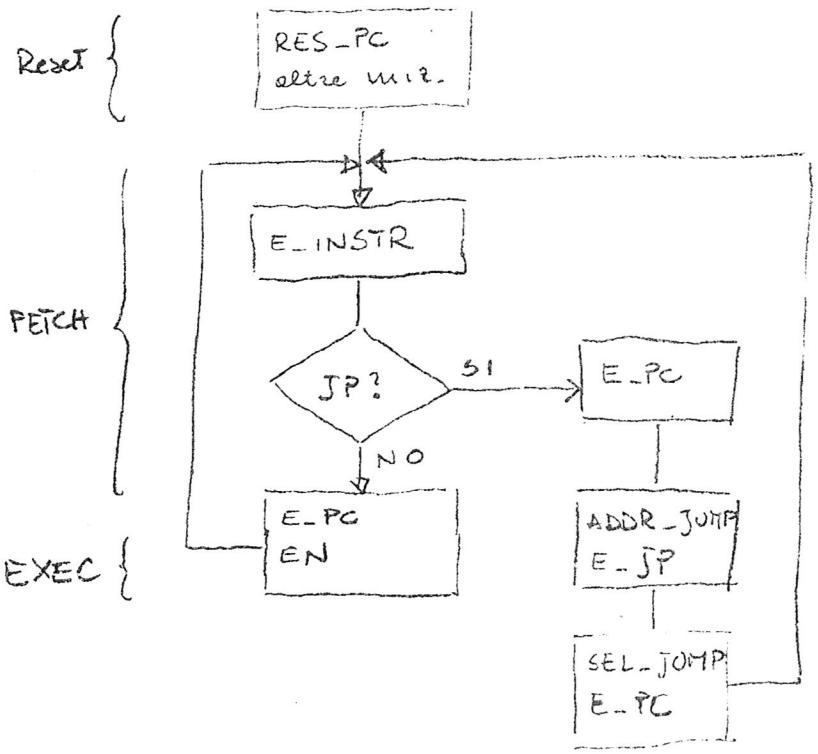
> Nella parte OPERATIVA occorre aggiungere un registro (FLAG) per memorizzare il valore dei bit Z e C dopo le operazioni aritmetiche.



> Ricapitolando la nuova configurazione per poter proseguire il sequenziatore



- Possibile microcodice



- Macchina a 6 stati
  Una istruzione "normale" richiede 2 CLOCK
- Il salto ne richiede 4

- Vediamo il programma assembler che realizza il precedente contatore

21.11

8 byte

byte	ADDR	ISTR	
1	loop:	LDI R0, 5	
1	loop1:	LDI R2, 11	
1		INC R0	
1		XOR R2, R0	; confronto con 11
2		JP z, loop	
2		JP loop1	

- Tempo di esecuzione del ciclo 70 cicli di clock

TRACCIA

CLK			
0	LDI R0, 5	R0 = 5	
2	LDI R2, 11	R2 = 11	
4	INC R0	R0 = 6	
6	XOR R2, R0	R2 = 11 XOR 6 ≠ 0	
8	JP z, loop	non eseguite	
12	JP	eseguite	
14	LDI R2, 11	R2 = 11	Ripete
16	INC R0	R0 = 7	
18	XOR R2, R0	R2 = 11 XOR 7 ≠ 0	
20	JP z, loop		
24	JP		
⋮			
60	JP		
62	LDI R2, 11		
64	INC R0	R0 = 11	
66	XOR		
70	JP z, loop	esegue	

21.12

- Set di istruzioni di un μP
  - > Caricamento e spostamento dati
    Registri, memoria, I/O
  - > Esecuzione di operazioni
    logiche, aritmetiche, spostamento di bit, selezione di bit
  - > Controllo del flusso di esecuzione
    Salti, sottoprogrammi

- Architetture CISC/RISC

> Set di istruzioni COMPLESSO	Mappare capacità operative Istruzioni "lente"
> Set di istruzioni RIDOTTO	Poche istruzioni veloci Individuare le più usate e ottimizzarle

- Architetture Harvard / Von Neumann