Istruzioni

- **Tipo R**

codop	(7)
rd	(5)
funz 3	(3)
rs1	(5)
rs2	(5)
funz 7	(7)

Add, Sub, And, ...

- **Tipo Immediate**

codop	(7)
rd	(5)
funz 3	(3)
rs1	(5)
Imm	(5+7 = 12)
	[-2048, +2047]

Addi

- **Tipo "S"**

Codop	(7)
imm	(5)
funz 3	(3)
rs1	(5)
rs2	(5)
funz 7	(7)

Solo istr. store

- **Tipo "SB"**

Codop	(7)
imm	(5)
funz 3	(3)
rs1	(5)
rs2	(5)
imm	(7)

Es: beq, bne, blt, bgt
bge ble

- **Tipo "J"**

Codop	(7)
rd	(5)
imm	(20)

Es: Jal

- **Tipo "U"**

Codop	(7)
rd	(5)
imm	(12)

Es: lui

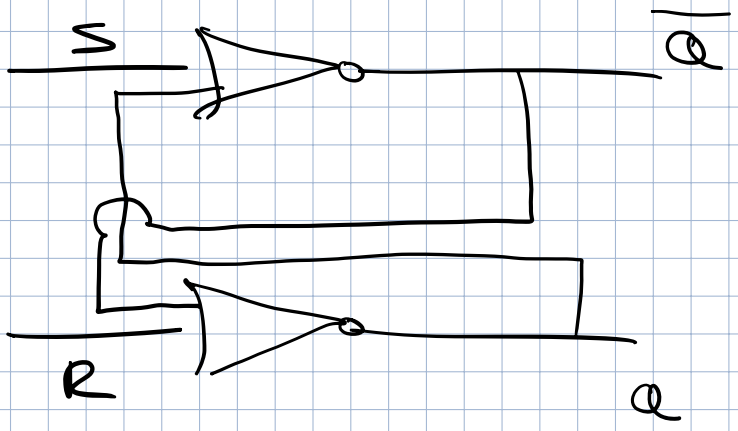
↳ Cambia che le istr. J hanno immediato sparso.

CODICI ALU

$\overline{A}$	$\overline{B}$	S			OPERAZIONE
0	0	0	0	⇒	AND
0	0	0	1	⇒	OR
0	0	1	0	⇒	SUM
0	1	1	0	⇒	SUB
0	1	1	1	⇒	LESS (a<b → a-b<0)
1	1	0	0	⇒	NOR

CIRCUITI SEQUENZIALI

- LATCH S-R S = Set 1 R = Reset 0
  Hold

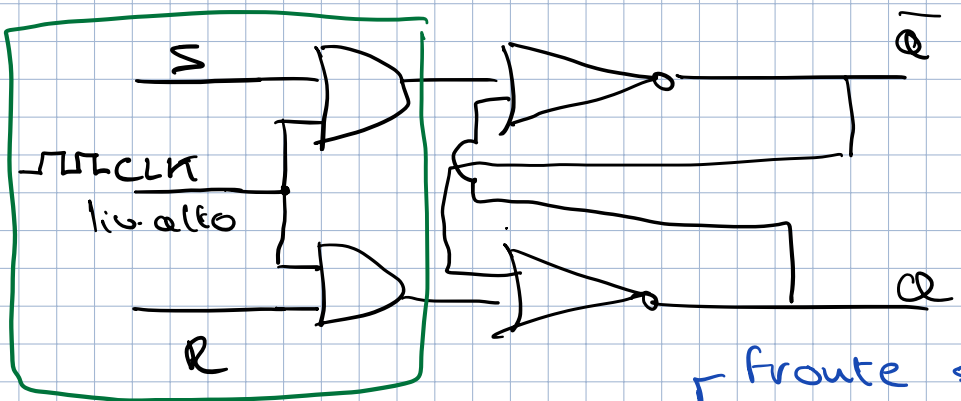


Combinazioni

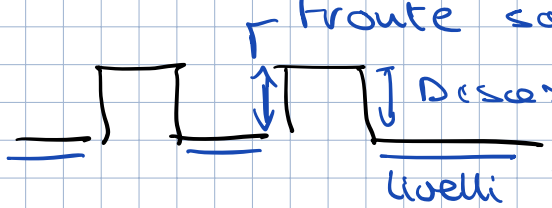
0 0 = Hold
1 0 = Set
0 1 = Reset
↑S ↑R

Problemi
- Combo 11 instabile ⟶ Verso Latch D
- Non è sincronizzato ⟶ Verso Agg. clock

- LATCH SR SINCRONIZZATO

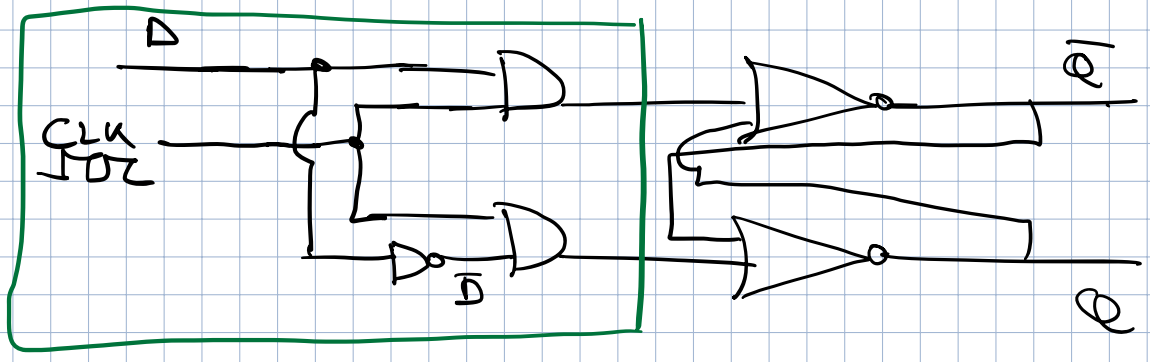


- Quando il CLK=0 equivale a Hold
- Rimane ancora instabilità



CIRCUITI SEQUENZIALI

- LATCH TIPO "D" SINCR.

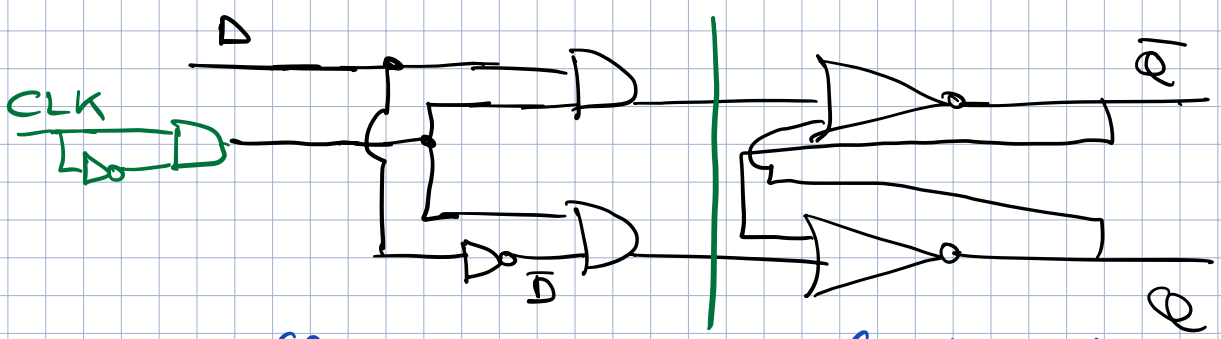


Combinaz.
- 0: Reset
- 1: Set

Clock: 0 → hold

  - Elimina S ed R sostituendole con D
  - Rimane il "Problema della concorrenza" con stati "ricorsivi" (es. add x_1, x_1, x_5) il valore oscilla. → Si risolve agg. una condizione imposs. in teoria

- LATCH TIPO "D" SINCR. A F. DI SALITA



  - Eff. non sta a fronte di salita ma riduce al minimo il livello alto.

ESEMPIO DI RISOLUZIONE ESER. CON MASK

Scrivere il codice RISC-V che restituisce sul **registro t2** il numero di bit uguali a 1 contenuti nel valore binario presente nel **registro t0**. Per esempio, se **t0** ha il valore binario equivalente al numero intero 37, il risultato atteso è 3. Suggerimento: usare opportunamente le istruzioni logiche and, srli etc.

Answer:

```
# t2 => ris da t0


li t3, 1
li t4, 64
li t2, 0
do_loop:
blt t4, zero, end_loop
    and t5, t0, t3
    add t2, t2, t5
    srli t0, t0, 1
    addi t4, t4, -1
    beq zero, zero, do_loop
end_loop:
```

Verifica risposta

L'esercizio chiede di (Dato un valore in un Reg
- Contare il numero di bit

Il modo migliore per gestire la cosa sarebbe controllare bit per bit e sommare.

ES: usiamo l'AND

t2 = 0 0 0 1 0 1 m= maschera

AND

m = 0 0 0 0 0 1

0 0 0 0 0 1 ✓ = 1_{Dec}

t2 = 0 0 0 1 0 0

AND

m = 0 0 0 0 0 1

0 0 0 0 0 0 ✓ = 0_{Dec}

Con questa mask viene controllato solo 1 bit (l'ultimo) → manca Iterare

Ogni registro in RISC V è formato da 64 bit quindi bisogna iterare 64 volte!

Manca capire come "muovere" il bit corrente

Basta usare l'istruzione srli che farà lo shift a destra (r).

Siccome è bit per bit va impostato a 1

srli $x_1, x_2, 1$

ASM: GESTIONE STRINGHE

- **ASSEGNAZIONE RAPIDA**

  Si può usare li per assegnare caratteri

```
li a1, '9'
li a2, '0'
```

- **GESTIONE TIPO "STRING"**

  Il tipo string non è altro che un'array di bytes.

  Per caricare il singolo char => lbu

```
str: .string "Hello World!"
```

```
lbu a2, 0(a0)
```

  (in a0 ci sta l'address)

- **SHIFT IN C** << oppure >>

  -> << : shift a sinistra sll

  -> >> : // // destra srl

```
hash << 5
```

```
slli a4, a1, 5      # hash << 5
```

- **ITERAZIONE**

  L'iterazione finisce quando il carattere deref == 0

```
lbu a2, 0(a0)

do_while:
beq a2, zero, end_while
```

PROBLEMI CON PRESTAZIONI

- # istr. al secondo = $\frac{\text{freq}}{CPI}$

Si considerino tre diversi processori P1, P2 e P3 che eseguono lo stesso insieme di istruzioni.

- P1 ha una frequenza di clock di 3 GHz e un CPI (Cicli Per Istruzione) di 1,5.
- P2 ha una frequenza di clock di 2,5 GHz e un CPI di 1,0.
- P3 ha una frequenza di clock di 4,0 GHz e un CPI di 2,2.
- Quale processore ha le prestazioni migliori espresse in numero di istruzioni al secondo?

$\frac{3}{\frac{3}{2}} = 3 \cdot \frac{2}{3} = \frac{6}{3} = 2$ P1 $\qquad \frac{2.5}{1} = 2.5$ P2 ✓ $\qquad \frac{4.0}{2.2} = 2.2$ P3

- # di cicli di clock più alto dato tempo

$V_s = \frac{\#\text{ clock}}{\text{freq}} \Rightarrow V_s \cdot \text{freq} = \#\text{clock}$

- Determinare il processore (scelto fra i 3 riportati) che necessita del numero di cicli di clock più alto, supponendo che tutti eseguano un programma in 10 secondi.

$P_1 \Rightarrow 10 \cdot 3 = 30 \qquad P_2 = 10 \cdot 2.5 = 25$

$P_3 \Rightarrow 10 \cdot 4 = 40$

- freq. di clock

$V_s = \frac{\#\text{ clock}}{\text{freq}} \Rightarrow \text{freq} = \frac{\#\text{ clock}}{V_s}$

- Si vorrebbe ridurre il tempo di esecuzione del 30% (da 10 a 7 secondi), ma per raggiungere questo obiettivo il CPI aumenterebbe del 20%. Quale sarebbe la frequenza di clock che consentirebbe questa riduzione del tempo di esecuzione per il processore P1? [Risposta espressa in GHz] 9

$P_1 = \quad V_s = 10 \qquad \#\text{clock} = 30 \qquad \text{freq} = \frac{30}{10} = 9$

$\frac{10}{100} + \frac{20}{100} \cdot 2 + \frac{50}{100} \cdot 3 + \frac{20}{100} \cdot 3$

$= \frac{10}{100} + \frac{40}{100} + \frac{150}{100} + \frac{60}{100} = \frac{260}{100} = 2.6$