Esercizi sulla progettazione di processori RISC

Paolo Romano

Progettare il Sottosistema di Calcolo (SCA) di un processore RISC, con organizzazione pipeline, con il minor numero di stadi, atto a supportare l'esecuzione delle seguenti istruzioni:

sintassi	*(semantica)*
sub $regdest, $regsorg1, $regsorg2	($regdest=$regsorg1 - $regsorg2)
add $regdest, $regsorg1, $regsorg2	($regdest=$regsorg1**+** $regsorg2)
subi $regdest, $regsorg, dato	($regdest = $regsorg – dato)
addi $regdest, $regsorg, dato	($regdest = $regsorg **+** dato)
load $regdest, indirizzo	($regdest = MEM[PC+indirizzo])
store $regsorg, indirizzo	(MEM[PC+4+indirizzo]=$regsorg)

Descrivere il formato delle istruzioni ipotizzando che:

- siano visibili 128 registri di dimensione 64 bits
- non ci siano altre istruzioni
- il sottosistema di memoria lavori con indirizzi a 64 bits
- il numero di bit a disposizione per l'istruzione sia 64

Formato Istruzione

- La definizione del formato istruzione influenza profondamente le successive scelte progettuali:
  - è buona regola massimizzare la regolarità del formato istruzioni per semplificare il progetto hardware

Formato Istruzione: Codifica Opcodes e Registri

Per codificare n informazioni differenti sono necessari:

$$\sup[\log_2(n)]$$

(dove con sup(x), $x \in \Re$, si indica la parte intera superiore di x)

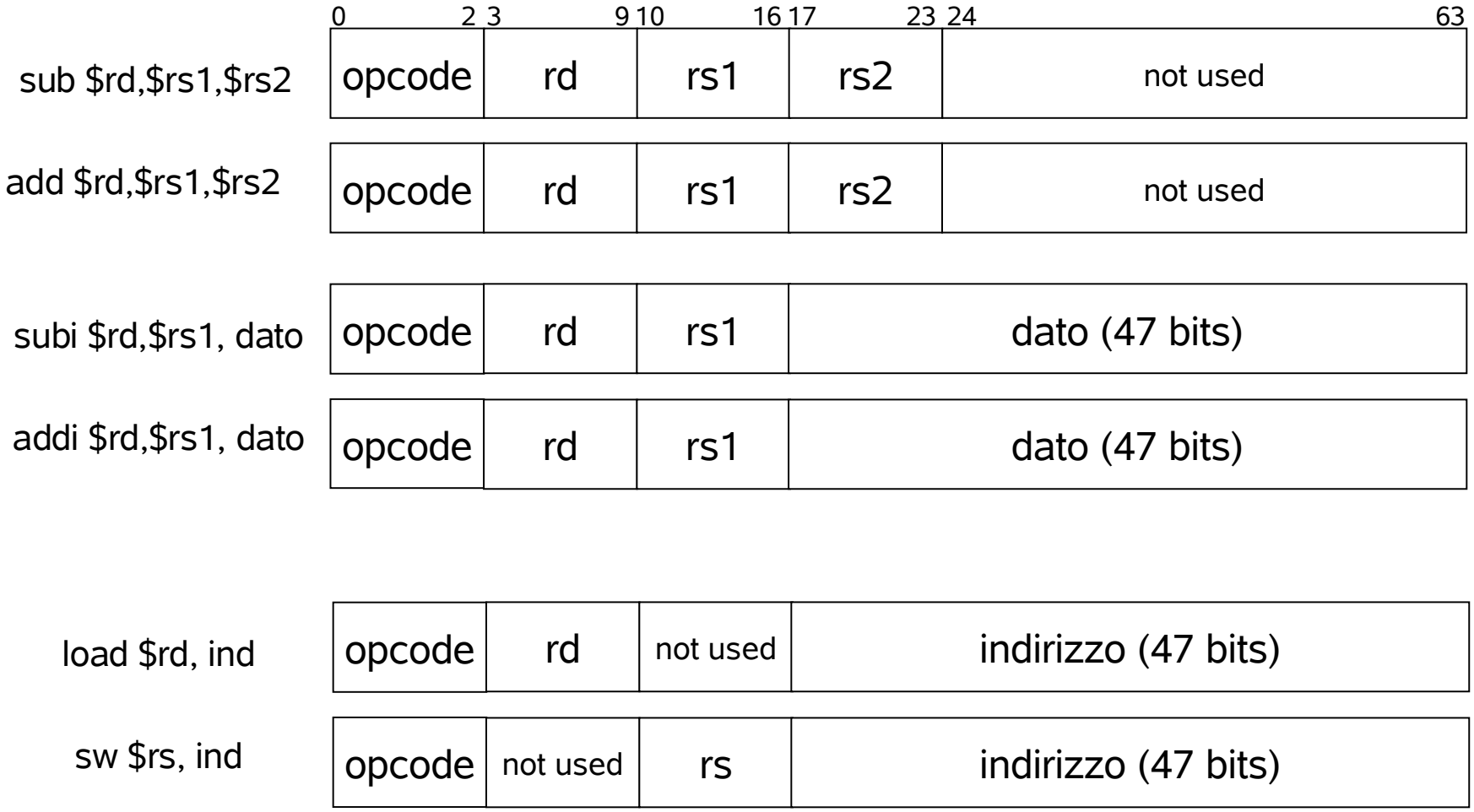
	opcode
sub $rd,$rs1,$rs2	000
add $rd,$rs1,$rs2	001
subi $rd,$rs1, dato	010
addi $rd,$rs1, dato	011

6 istruzioni $\Rightarrow$ 3 bits

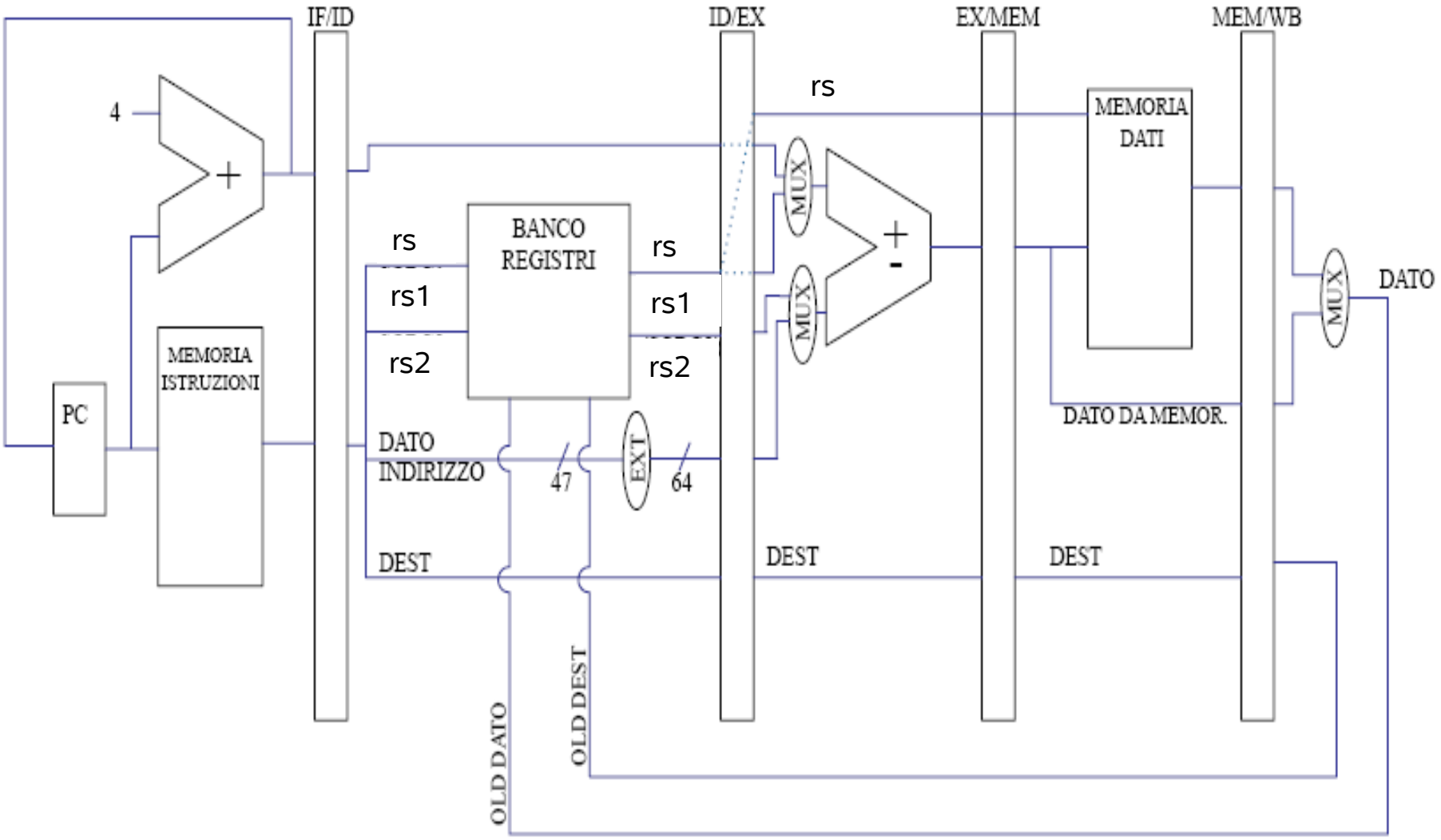
128 registri $\Rightarrow$ 7 bits

load $rd, ind	100
sw $rs, ind	101

Formato Istruzione

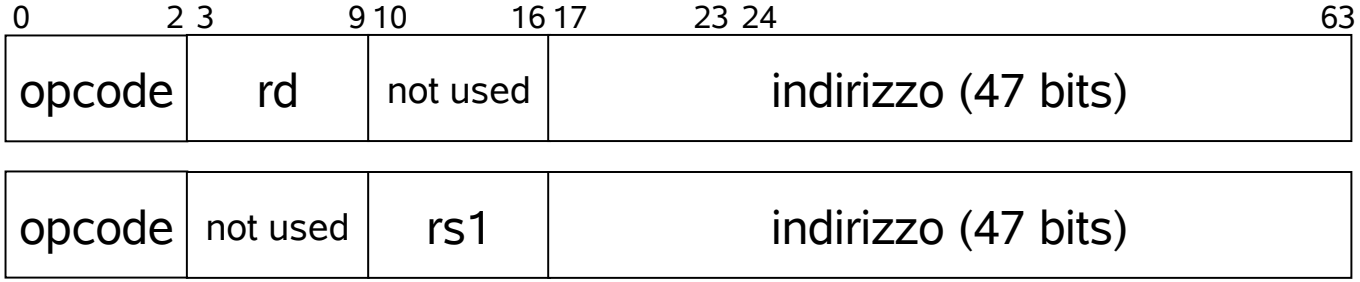


Progettazione SCA

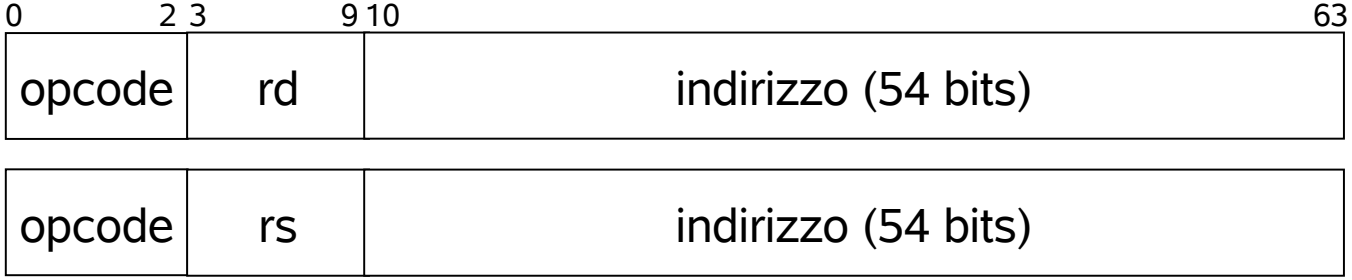


E se avessimo optato per un formato istruzione + irregolare?

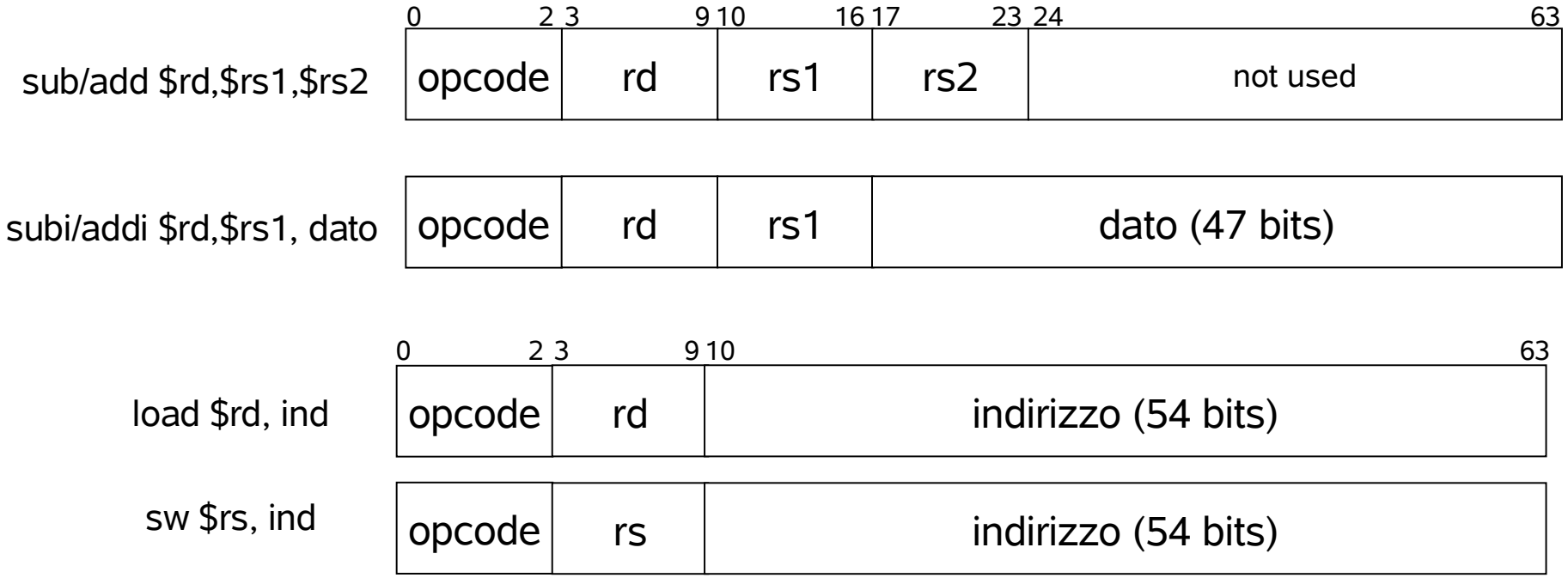
- Nel precedente formato le istruzioni di load e store lasciano inutilizzato rispettivamente il campo rd e rs1:



- Pro:
  - i bits [3-9] e [10-16] sono rispettivamente associati a indirizzi di registri da scrivere/leggere, indipendentemente dall'istruzione
- Contro:
  - il numero di bits a disposizione per la codifca dell'indirizzo è minore del max teorico, ovvero 54:



E se avessimo optato per un formato istruzione + irregolare?



Una simile scelta rende più complesso determinare quali bits dell'istruzione usare per indirizzare i registri da leggere durante la fase di ID:

- l'istruzione sw utilizza i bits [3-9] per specificare il registro da leggere, a differenza di sub/add e subi/addi che usano i bit [10-17] e [17-23]

Instruction set meno regolare = HW più complesso