Università degli Studi di Milano «La Statale»
Dipartimento di Informatica

Lezione 12:

CPU e linguaggio MIPS

Parte F: la jump register

Marco Tarini

165

Un'altra istruzione MIPS : **jump register**

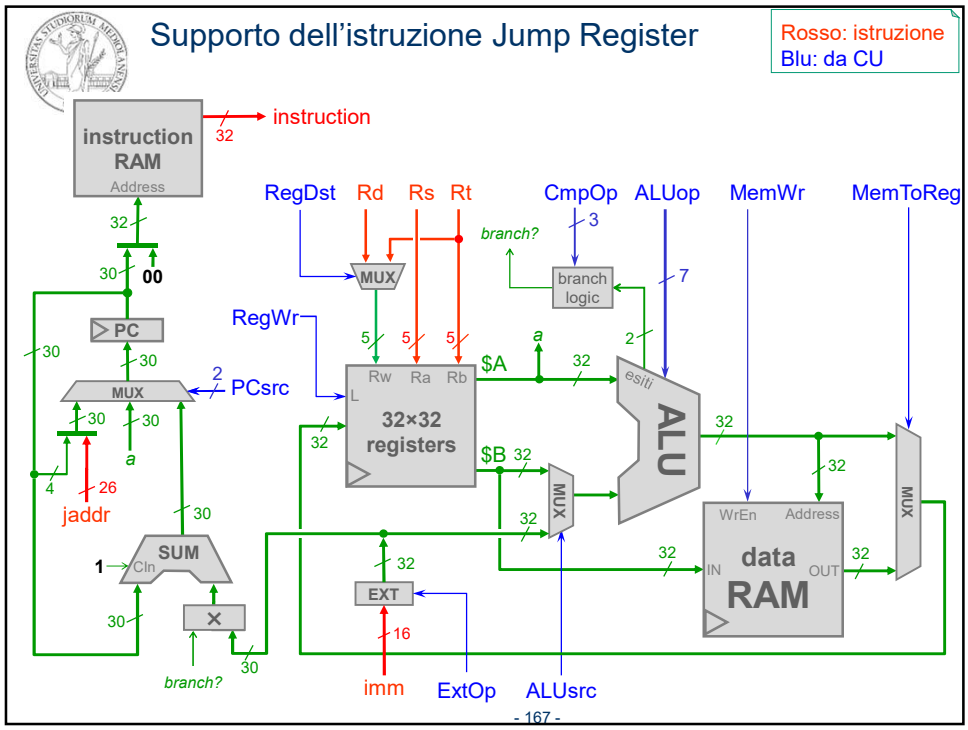
0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Operaz su registri	6	---	---	---	Jump register
OPCODE	*RS*	*RT*	*RD*	*---*	*FUNCTION*

Tradotta in assembly MIPS: `JR $6`

A parole: *« Salta all'istruzione che sta all'indirizzo contenuto nel Registro 6»*

166



167

Supporto dell'istruzione Jump Register

E' relativamente banale aggiungere al datapath il supporto di questa istruzione:

- Aggiunta al MUX che controlla l'ingresso del PC di una 3za opzione:
  - il registro $a dal banco dei registri
  - per la precisione, solo i suoi 30 bit più significativi
  - (nota che al valore del PC viene sempre aggiunto 00 come i due bit meno significativi)
- Il selezionatore è ora a 3 ingressi, e richiede 2 bit di controllo ($3 \leq 2^2$)
  - Il comando di controllo PCsrc passa a 2 bit
  - Dovrà valere 00, 01, o 10 per le tre opzioni

- 168 -

168

Sommario: i tre tipi di salti (modifica del PC) in MIPS

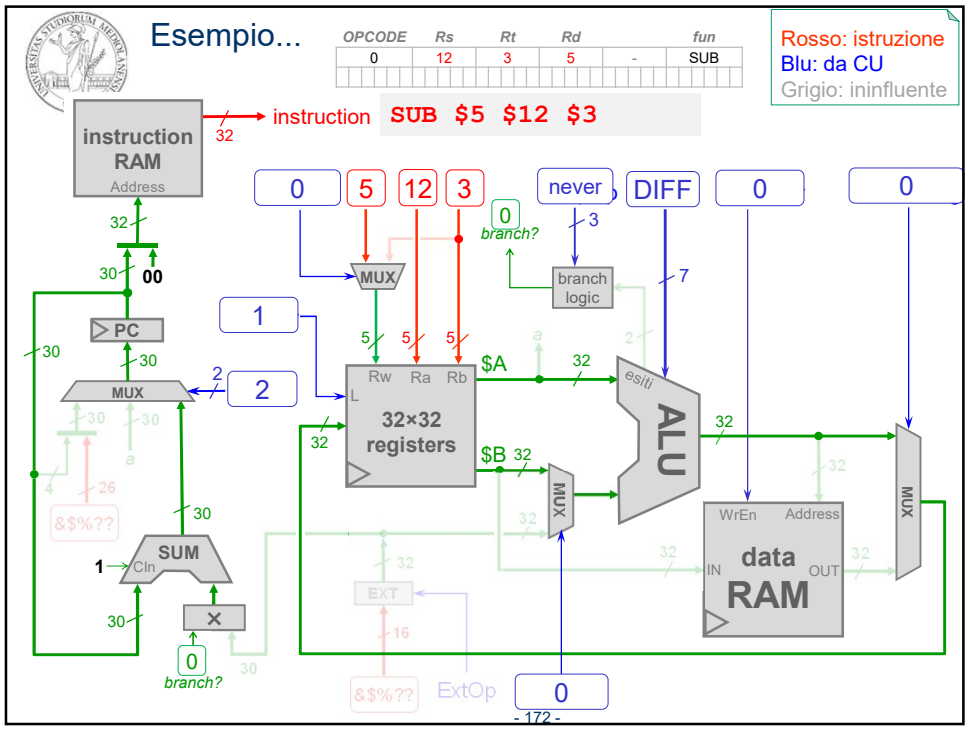
- **BRANCH**
  - Salto **condizionato** (biforcazione, bivio)
    (salta «solo se…», altrimenti, passa alla istruz successiva)
  - Limite: salto breve (max 32K istruzioni avanti o indetro)
  - Indirizzo **relativo**
    (destinazione definita come differenza rispetto al PC corrente)
- **JUMP *address***
  - Salto **non condizionato** (salta sempre)
  - Indirizzo **assoluto**
    (il campo address ridefinisce il PC, sovrascrivendo quello corrente)
  - Limite: salta solo **entro il «blocco»**
    (cioè non può modificare i primi 4 bit del PC)
- **JUMP *register***
  - Salto **non codizionato**, **assoluto**, niente limiti (1 registro = 32 bit)
  - Prezzo: bisogna prima approntare un registro con la destinazione

169

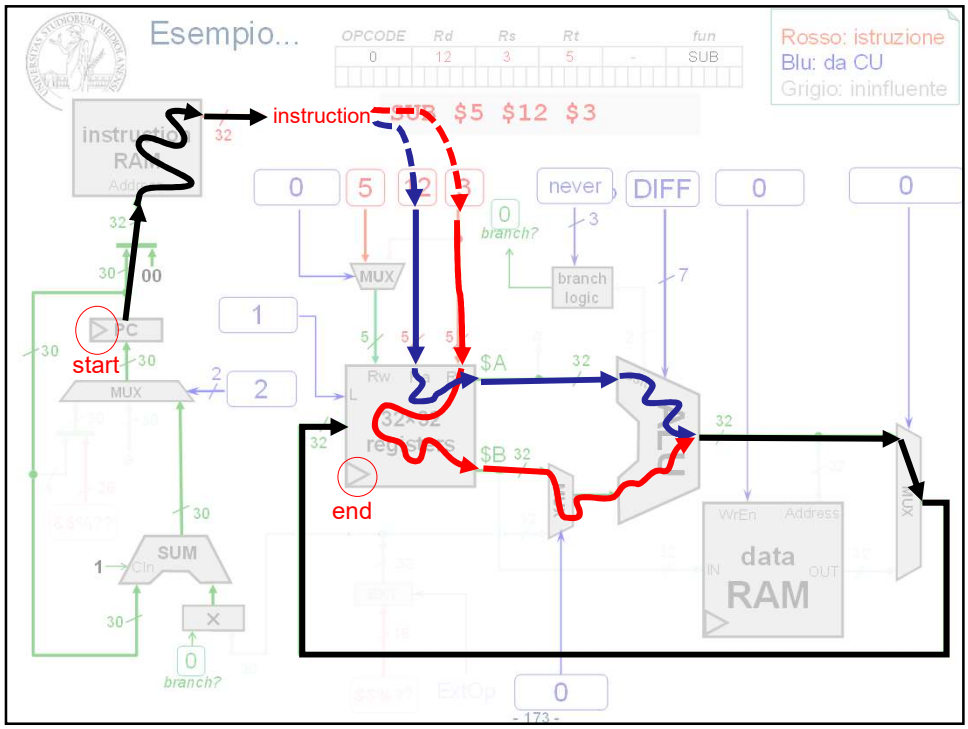
Il datapath è concluso.

- Considereremo il datapath visto fin'ora *concluso*
- Verifichiamo che è ancora in grado di eseguire la prima istruzione che avevamo supportato: *l'operazione aritmetica fra due registri*
  - Nonostante abbiamo aggiunto il supporto
    a molti altri tipi di istruzione!
  - Proveremo ad esempio con una la «sub» (sottrazione)
  - Quali comandi è necessario che siano mandata dalla Control Unit per eseguirla? Vediamo
- Ricorda: nei disegni, usiamo sempre la convenzione che nei MUX i canali ingressi sono numerati (partendo da 0)…
  - …da sinistra a destra, o
    …da l'alto verso il basso!

171


Esempio...
OPCODE Rs Rt Rd fun
0 12 3 5 - SUB
Rosso: istruzione
Blu: da CU
Grigio: ininfluente
instruction
SUB $5 $12 $3
instruction RAM
Address
32
0 5 12 3
never
DIFF
0
0
branch?
branch logic
MUX
PC
00
1
2
Rw Ra Rb
32×32 registers
$A
$B
ALU
esiti
WrEn Address
data RAM
IN OUT
SUM
Cin
EXT
ExtOp
- 172 -

172


Esempio...
Rosso: istruzione
Blu: da CU
Grigio: ininfluente
instruction
start
end
- 173 -

173

Datapath e durata del ciclo di clock

- Osserva il datapath dopo che hai diramato
  - l'istruzione (campo per campo)
  - i comandi dalla CU
- Allo scattare del ciclo di clock (fronte in discesa)…
  - Il registro PC assume il suo nuovo valore (che attendeva nel suo ingresso)
- Questo determina tutto il resto, in cascata! Cioè:
  - il valore del PC viene presentato in input (address) alla RAM istruzioni
  - Questa RAM (dopo il suo tempo di commutazione – la fase di fetch) presenta in uscita l'istruzione corrente
  - Questa istruzione viene diramata al resto del dadapath e in input alla CU
  - La CU (un circuito) produce i comandi, che sono diramati ai vari componenti …
  - Etc… cioè la fase execute (vedi schema)
- Fino a giungere al momento in cui (in questo caso) al banco dei registri viene presentato un nuovo valore da memorizzare
  - Questi dati vengono memorizzato allo scattare del ciclo di clock successivo,
  - (guai, se per allora non sono ancora pronti!)
- Il **ciclo di clock** deve durare abbastanza da consentire a tutto il procedimento (quello fra i due ▷) di avvenire fra i 2 fronti in discesa (che delimitano il ciclo di clock)

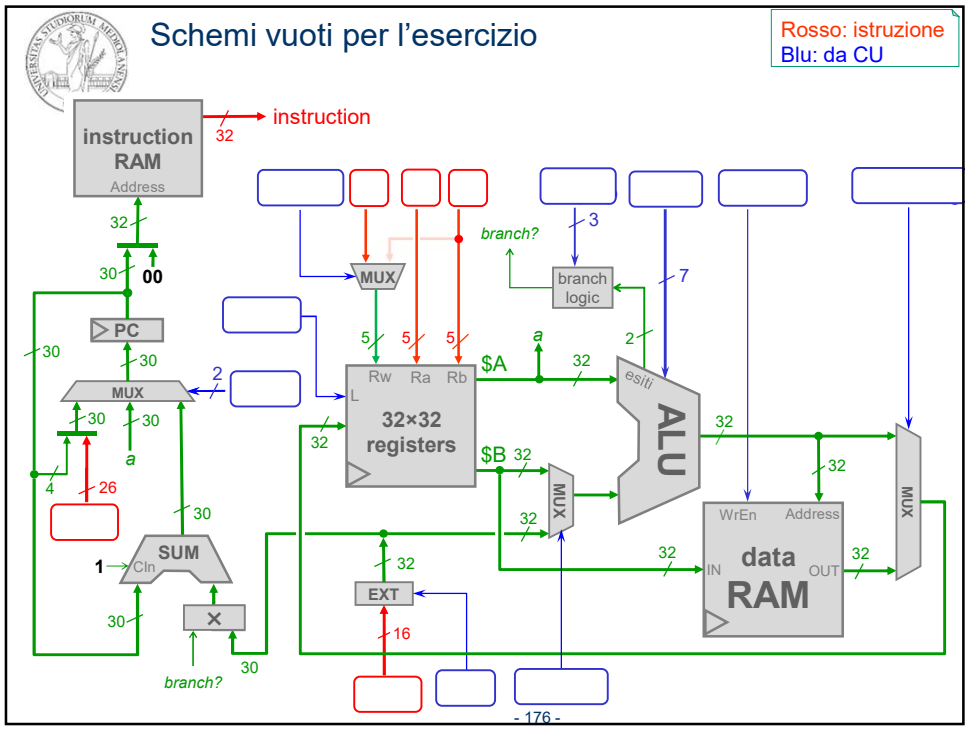
Esercizio sul datapath

- Determina, nello stesso modo appena visto, i segnali di comando che devono essere prodotti dalla CU,
  e tutti i valori dei campi dell' istruzione diramati alla ALU,
  per le seguenti istruzioni:

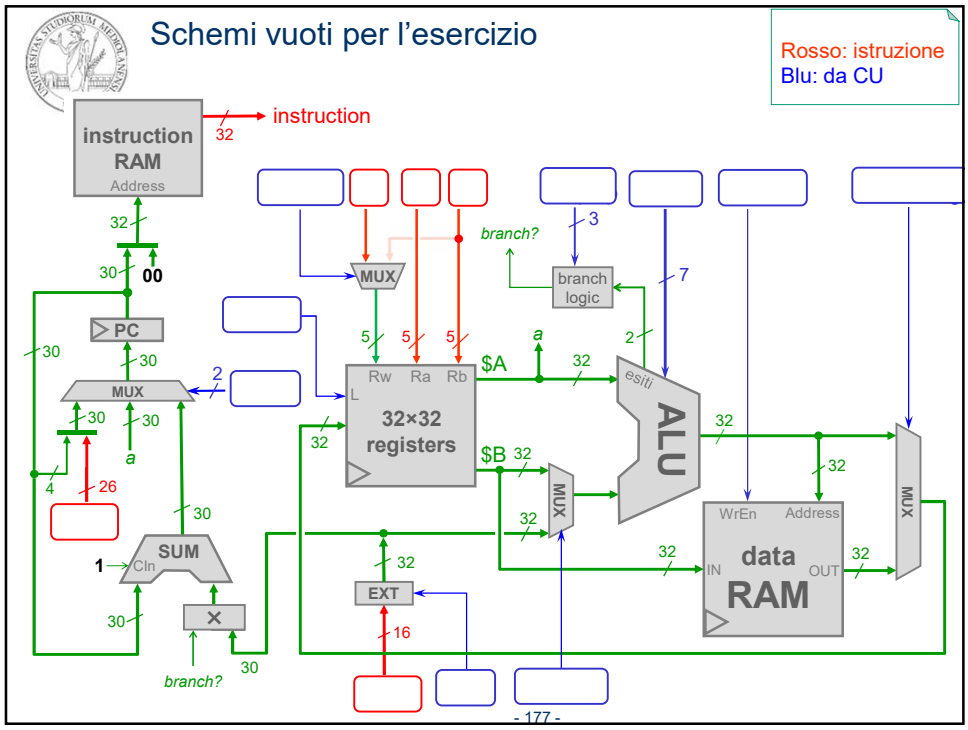
```
SW $23 12($9)
LW $23 12($9)
BEQ $4 $9 -20
J 0x1234000
```

- Determina, in ciascun caso, quale/i elemento sincrono (▷) deve ricevere il proprio input stabile prima del concludersi del ciclo di clock, da memorizzare
- Accertati anche di saper tradurre i 4 comandi in linguaggio naturale (l'Italiano, per es)

(chi volesse fare questo esercizio con carta e penna può stampare
la pagina successiva tante volte quante necessario)


Schemi vuoti per l'esercizio
Rosso: istruzione
Blu: da CU
instruction RAM
Address
instruction
32
PC
MUX
00
branch?
branch logic
3
7
2
5
5
5
Rw
Ra
Rb
32×32 registers
$A
$B
a
32
esiti
ALU
WrEn
Address
data RAM
IN
OUT
SUM
Cin
1
4
26
30
EXT
16
- 176 -

176


Schemi vuoti per l'esercizio
Rosso: istruzione
Blu: da CU
instruction RAM
Address
instruction
32
PC
MUX
00
branch?
branch logic
3
7
2
5
5
5
Rw
Ra
Rb
32×32 registers
$A
$B
a
32
esiti
ALU
WrEn
Address
data RAM
IN
OUT
SUM
Cin
1
4
26
30
EXT
16
- 177 -

177