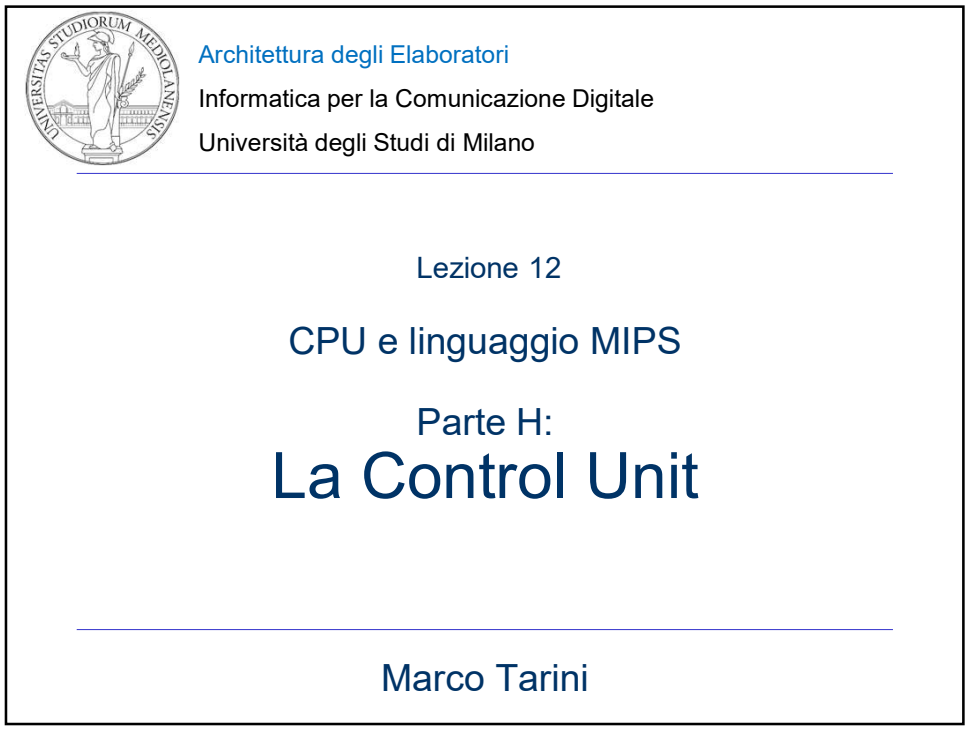


Architettura degli Elaboratori

Informatica per la Comunicazione Digitale

Università degli Studi di Milano

Lezione 12

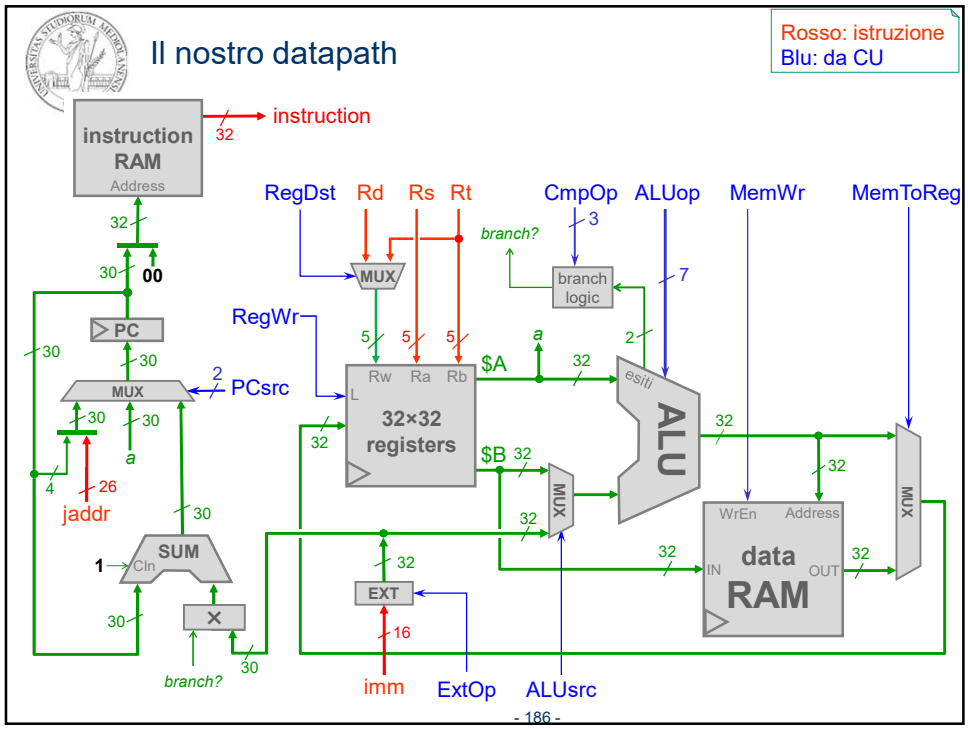
CPU e linguaggio MIPS

Parte H:

La Control Unit

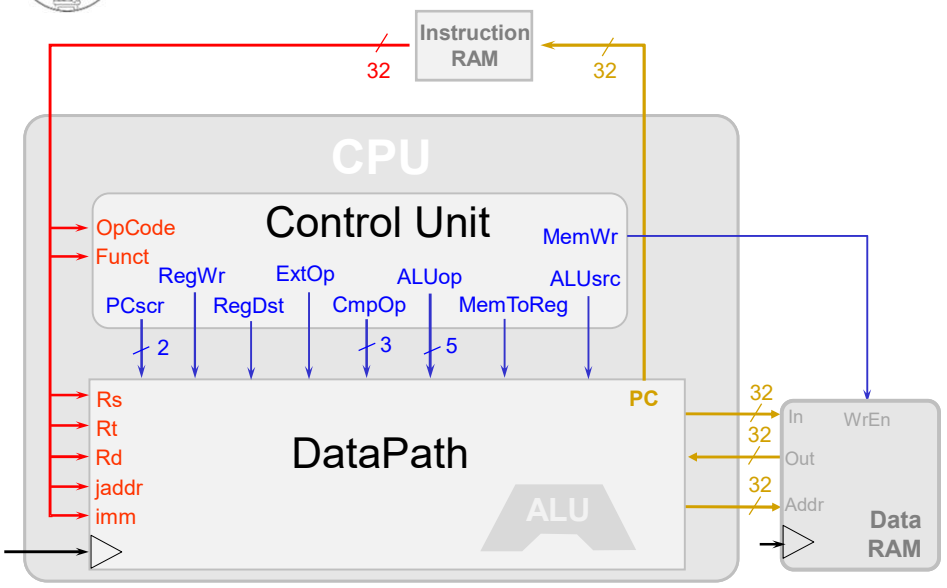
Marco Tarini

185



186

La CPU vista a più ad alto livello. Organizzazione in chip di un micropocessore



- 187 -

187

Quadro Complessivo a più alto livello: note

- Rs, Rt, Rd, imm e jaddr sono passati a datapath
- OpCode e Funct sono passati alla Control Unit
- La Control Unit (CU) produce i controlli che guidano il datapath
  - che è un circuito combinatorio
  - il lavoro della CU è la fase di "interpretazione dell'istruzione" del ciclo fetch and execture
- A causa del formato variabile delle istruzioni MIPS, alcuni bit dell'istruzione sono replicati in campi diversi contemporaneamente; per es:
  - i 5 bit di Rd sono anche la prima parte dei 16 bit di Imm,
  - i 5 bit di Funct sono anche l'ultima parte dei 16 bit di Imm,
  - i 26 bit di Jaddr sono gli stessi che compongono tutti gli altri campi

  …ma i controlli mandati dalla CU fanno sempre in modo che i campi senza senso (per es, Rd in una istruzione di tipo I) non abbiano alcun effetto (per es, siano nella parte ignorata di un selezionatore)

- 188 -

188

L'Istruction Set MIPS

Primi sei bit dell'istruzione

Ultimi sei bit dell'istruzione

Mnemonic	Meaning	Type	OpCode	Funct
add	Add	R	0x00	0x20
addu	Add Unsigned	R	0x00	0x21
and	Bitwise AND	R	0x00	0x24
div	Divide	R	0x00	0x1A
divu	Unsigned Divide	R	0x00	0x1B
jr	Jump to Address in Register	R	0x00	0x08
mfhi	Move from HI Register	R	0x00	0x10
mthi	Move to HI Register	R	0x00	0x11
mflo	Move from LO Register	R	0x00	0x12
mtlo	Move to LO Register	R	0x00	0x13
mult	Multiply	R	0x00	0x18
multu	Unsigned Multiply	R	0x00	0x19
nor	Bitwise NOR (NOT-OR)	R	0x00	0x27
xor	Bitwise XOR (Exclusive-OR)	R	0x00	0x26
or	Bitwise OR	R	0x00	0x25
slt	Set to 1 if Less Than	R	0x00	0x2A
sltu	Set to 1 if Less Than Unsigned	R	0x00	0x2B
sll	Logical Shift Left	R	0x00	0x00
srl	Logical Shift Right (0-extended)	R	0x00	0x02
sra	Arithmetic Shift Right (sign-extended)	R	0x00	0x03
sub	Subtract	R	0x00	0x22

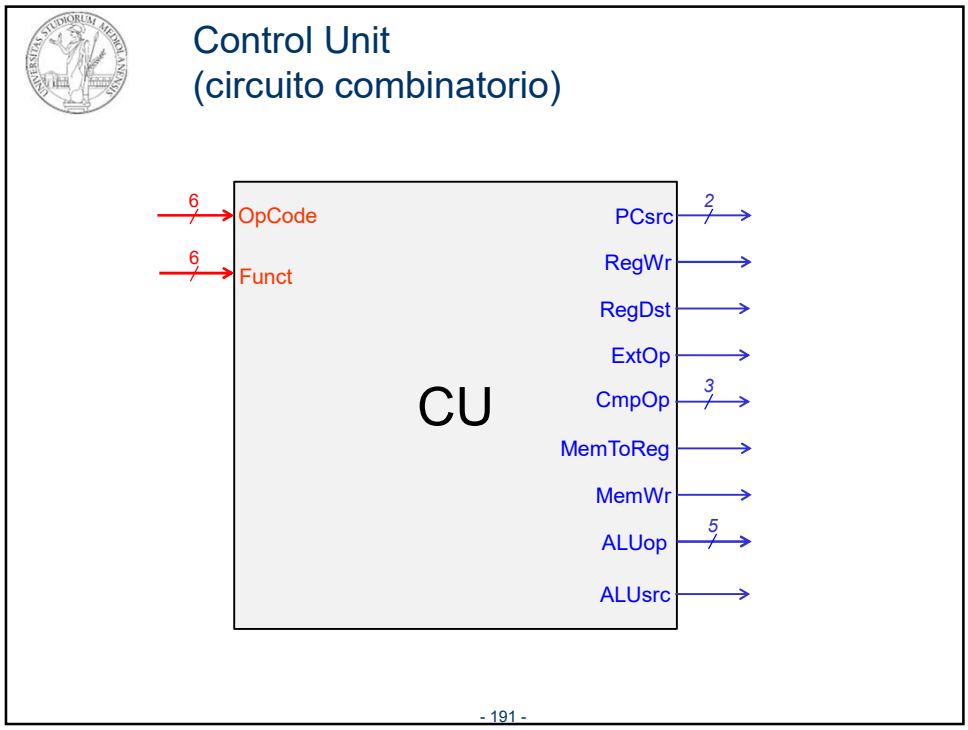
189

subu	Unsigned Subtract	R	0x00	0x23
slti	Set to 1 if Less Than Immediate	I	0x0A	NA
sltiu	Set to 1 if Less Than Unsigned Immediate	I	0x0B	NA
andi	Bitwise AND Immediate	I	0x0C	NA
ori	Bitwise OR Immediate	I	0x0D	NA
lui	Load Upper Immediate	I	0x0F	NA
j	Jump to Address	J	0x02	NA
sw	Store Word	I	0x2B	NA
jal	Jump and Link	J	0x03	NA
beq	Branch if Equal	I	0x04	NA
bne	Branch if Not Equal	I	0x05	NA
blez	Branch if Less Than or Equal to Zero	I	0x06	NA
bgtz	Branch on Greater Than Zero	I	0x07	NA
addi	Add Immediate	I	0x08	NA
addiu	Add Unsigned Immediate	I	0x09	NA
mfc0	Move from Coprocessor 0	I	0x10	NA
lb	Load Byte	I	0x20	NA
lw	Load Word	I	0x23	NA
lbu	Load Byte Unsigned	I	0x24	NA
lhu	Load Halfword Unsigned	I	0x25	NA
sb	Store Byte	I	0x28	NA
sh	Store Halfword	I	0x29	NA

(non abbiamo visto tutte le istruzioni di questa lista)
(il nostro DataPath non le supporta tutte)

- 190 -

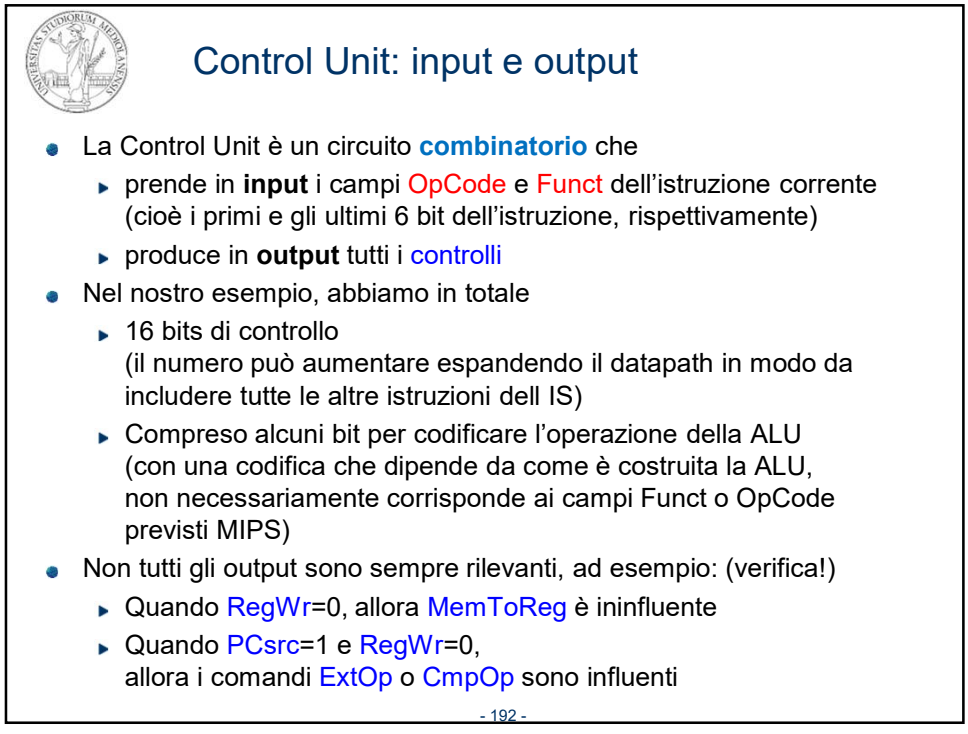
190



Control Unit
(circuito combinatorio)

- 191 -

191



Control Unit: input e output

- La Control Unit è un circuito **combinatorio** che
  - prende in **input** i campi OpCode e Funct dell'istruzione corrente (cioè i primi e gli ultimi 6 bit dell'istruzione, rispettivamente)
  - produce in **output** tutti i controlli
- Nel nostro esempio, abbiamo in totale
  - 16 bits di controllo (il numero può aumentare espandendo il datapath in modo da includere tutte le altre istruzioni dell IS)
  - Compreso alcuni bit per codificare l'operazione della ALU (con una codifica che dipende da come è costruita la ALU, non necessariamente corrisponde ai campi Funct o OpCode previsti MIPS)
- Non tutti gli output sono sempre rilevanti, ad esempio: (verifica!)
  - Quando RegWr=0, allora MemToReg è ininfluente
  - Quando PCsrc=1 e RegWr=0, allora i comandi ExtOp o CmpOp sono influenti

- 192 -

192

Control Unit: progettazione

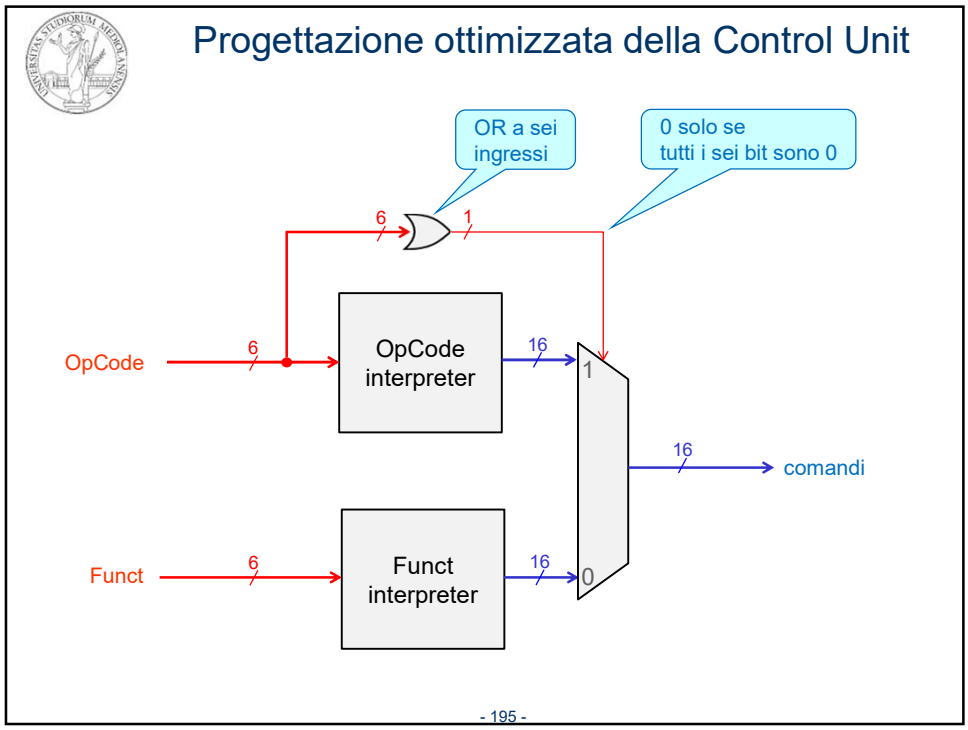
- Per progettare questo circuito, potremmo procedere come per qualsiasi circuito combinatorio, cioè «semplicemente» …
  - Scrivere la sua (lunga) tabella delle verità
  - Ogni sua riga corrisponde ad un tipo di istruzione (vedi tabella) e quindi ad un valore per ogni bit di controllo
  - Nota: questa tabella riporta anche i valori ininfluenti, come delle «X» («don't care»)
  - Trasformarla in un circuito, come abbiamo imparato (per es, tramite 1ma forma normale, come Somma di Prodotti
- Nel nostro esempio, questa tabella avrebbe
  - 2^{12} = 2048 righe
  - 16 colonne di output
- Non è impossibile procedere in questo modo, ma c'è un modo più pratico…

193

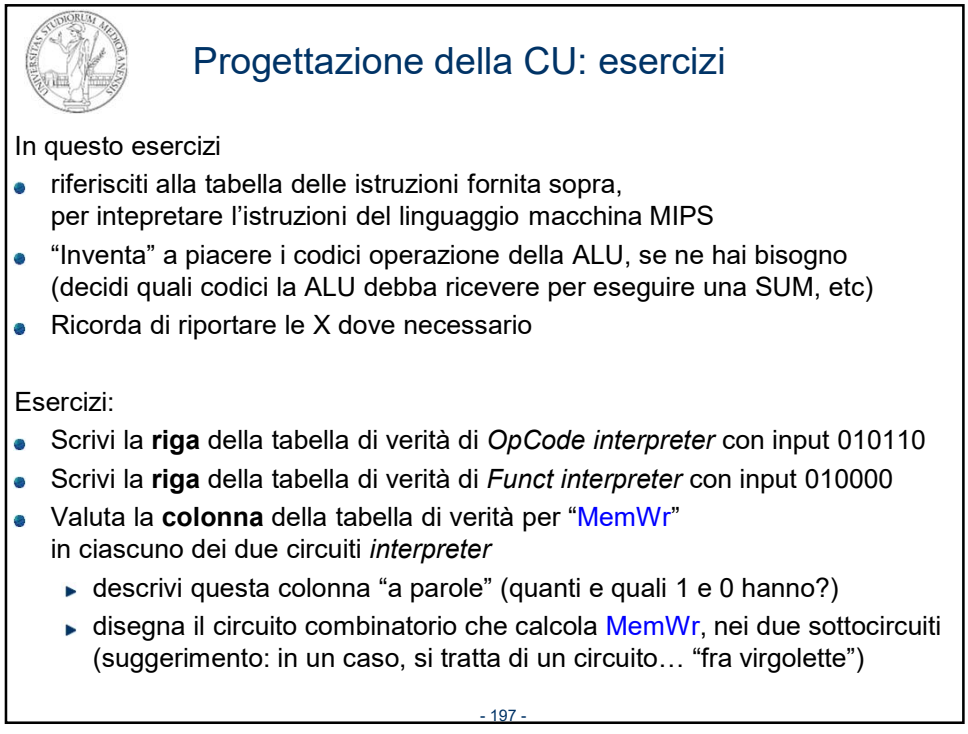
Control Unit: progettazione ottimizzata

- Come semplificazione, notiamo che le istruzioni si dividono in due gruppi:
- Istruzioni R
  - cioè tutte le operazioni fra registri (e anche **`jr`**, **`mfhi`**, e poche altre)
  - Hanno tutte OpCode = **000000**
  - L'output della CU dipende solo da Funct
  - Molti output sono costanti, per es, MemWr = 0
- Tutte le altre
  - Compreso: operazioni con immediate, le branch, le load, le store, …
  - Hanno OpCode ≠ **000000**
  - L'output della CU dipende solo da OpCode
  - il campo Funct, infatti, non è valido (è… parte di imm o di jaddr)
- Questa osservazione ci consente di dividere la CU in due sotto circuiti separati, ciascuno assai più semplice (tabella di, al max, 2^6 = 64 righe)…
  - non è un caso: MIPS-32 è un ISA progettato con molta attenzione alla sempicità di implementazione HW

194


Progettazione ottimizzata della Control Unit
OR a sei ingressi
0 solo se tutti i sei bit sono 0
6
1
OpCode
6
OpCode interpreter
16
1
16
comandi
Funct
6
Funct interpreter
16
0
- 195 -

195


Progettazione della CU: esercizi
In questo esercizi
riferisciti alla tabella delle istruzioni fornita sopra, per intepretare l'istruzioni del linguaggio macchina MIPS
"Inventa" a piacere i codici operazione della ALU, se ne hai bisogno (decidi quali codici la ALU debba ricevere per eseguire una SUM, etc)
Ricorda di riportare le X dove necessario
Esercizi:
Scrivi la riga della tabella di verità di OpCode interpreter con input 010110
Scrivi la riga della tabella di verità di Funct interpreter con input 010000
Valuta la colonna della tabella di verità per "MemWr" in ciascuno dei due circuiti interpreter
descrivi questa colonna "a parole" (quanti e quali 1 e 0 hanno?)
disegna il circuito combinatorio che calcola MemWr, nei due sottocircuiti (suggerimento: in un caso, si tratta di un circuito… "fra virgolette")
- 197 -

197