Esempio di architettura RISC: famiglia MIPS

- Riferimenti bibliografici: **Stallings + Hennessy & Patterson**
- Architettura molto regolare con insieme di istruzioni semplice e compatto
- Architettura progettata per una implementazione efficiente della pipeline (lo vedremo più avanti)
- Codifica delle istruzioni omogenea: 32 bit
- Co-processore per istruzioni a virgola mobile e gestione delle eccezioni

MIPS

Registri

- 32 registri di 32 bit (registro 0 contiene sempre il valore 0)
- Architettura Load / Store
  - Istruzioni di trasferimento per muovere i dati tra memoria e registri
  - Istruzioni per la manipolazione di dati operano sui valori dei registri
  - Nessuna operazione memoria ↔ memoria
- Quindi: le istruzioni operano su registri (registro i riferito con $i)
- Esempio: add $1, $2, $3

MIPS

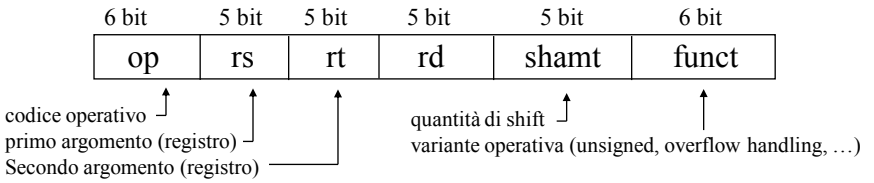
Dati e modi di indirizzamento

- Registri possono essere caricati con byte, mezze parole, e parole (riempiendo con 0 quando necessario o estendendo, cioè replicando, il segno sui bit non coinvolti del registro)
- Modalità di indirizzamento ammesse (con campi di 16 bit):
  - Immediata es. add $2, $2, 0004
  - Displacement es. sw $1, 000c($1)
- Altre modalità derivabili:
  - Indiretta registro (displacement a 0) es. sw $2, 0000($3)
  - Assoluta (registro 0 come registro base) es. lw $1, 00c4($0)

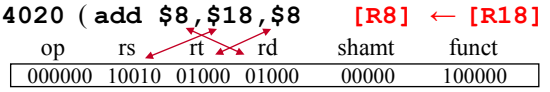
MIPS

Formato Istruzioni

- 32 bit per tutte, 3 formati diversi (formato R, formato I, formato J)
- **Formato R (registro)**



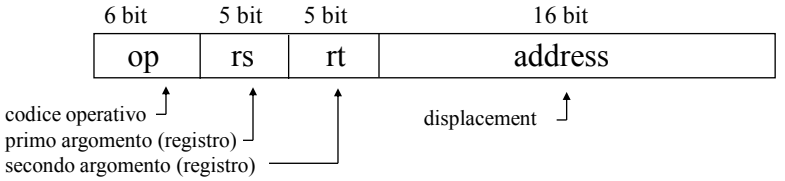
Es: `0x02484020` (`add $8,$18,$8` `[R8] ← [R18] + [R8]`)



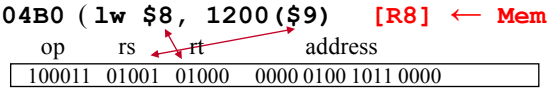
MIPS

Formato Istruzioni

- **Formato I (istruzioni load / store)**



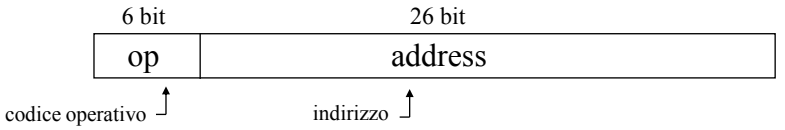
Es: **0x8D2804B0** (**lw $8, 1200($9)** **[R8] ← Mem[1200+[R9]]**)



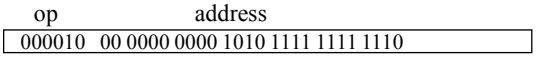
MIPS

Formato Istruzioni

- **Formato J (istruzioni jump)**



Es: **0x0800AFFE** (**j 45054** **[PC] ← 45054**)



Fasi (MIPS)

Fasi senza pipeline:

IF (istruction fetch):

- IR ← Mem[PC] ;
- NPC ← PC + 4 ;

Dove NPC è un registro temporaneo

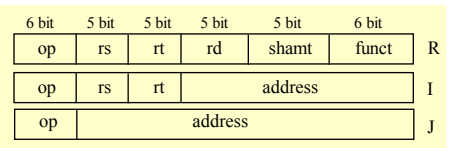
PC è il program counter

Fasi (MIPS)

ID (instruction decode/register fetch cycle):

- A ← Regs[rs] ;
- B ← Regs[rt] ;
- Imm ← campo immediato di IR con segno esteso ;

Dove A, B, Imm sono registri temporanei

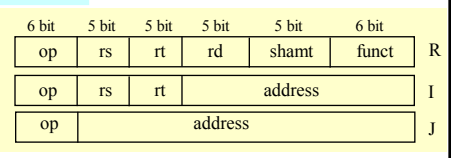


Fasi (MIPS)

EX (execution/effective address cycle):

1. ***Riferimento a memoria*** `lw $8, 1200($9)`
   - ALUOutput ← A + Imm ;
2. ***Istruzione ALU registro-registro*** `add $8,$18,$8`
   - ALUOutput ← A ***func*** B ;
3. ***Istruzione ALU registro-immediato*** `addi $8,$18,4`
   - ALUOutput ← A ***op*** Imm ;
4. ***Salto*** `j 45054`
   - ALUOutput ← NPC + (Imm << 2);
   - Cond ← (A = = 0) ;

shift a sinistra

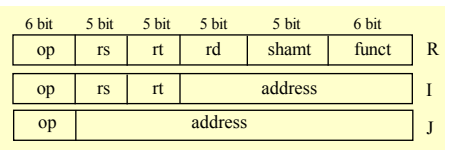


Fasi (MIPS)

MEM (memory access/branch completion cycle):

- PC ← NPC ; in tutti i casi

1. ***Riferimento a memoria*** `lw $8, 1200($9)`
   - LMD ← Mem[ALUOutput] or
     Mem[ALUOutput] ← B; `sw $5, 1700($4)`
2. ***Salto*** `j 45054`
   - if (Cond) PC ← ALUOutput ;



Fasi (MIPS)

WB (write/back cycle):

1. ***Istruzione ALU registro-registro*** `add $8,$18,$8`
   - Regs[rd] ← ALUOutput ;
2. ***Istruzione ALU registro-immediato*** `addi $8,$18,4`
   - Regs[rt] ← ALUOutput ;
3. ***Istruzione Load*** `lw $8, 1200($9)`
   - Regs[rt] ← LMD ;