The DLX ISA

DLX Instruction Set Architecture

Prof. G. Ascia

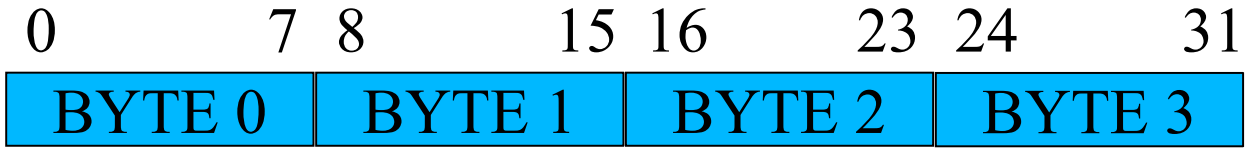
- **Concetti architetturali:**
  - Semplicità del load/store IS
  - Semplicità nella decodifica (istruzioni a lunghezza fissa)
- **Caratteristiche**
  - 32 General Purpose Registers
  - 32 Floating Point Registers a singola precisione (condivisi con 16 FPRs a doppia precisione)
  - La lunghezza di una word è di 32 bit.
  - Indirizzamento della memory a byte, Big Endian, indirizzi di 32

Registri

- L'ISA del DLX contiene 32 (R0-R31) general-purpose registers da 32 bit
- I registri R1-R31 sono dei reali GP registers
- R0 contiene sempre il valore 0 e non può essere modificato
- R31 è utilizzato per conservare l'indirizzo di ritorno per le istruzioni JAL e JALR

Registri

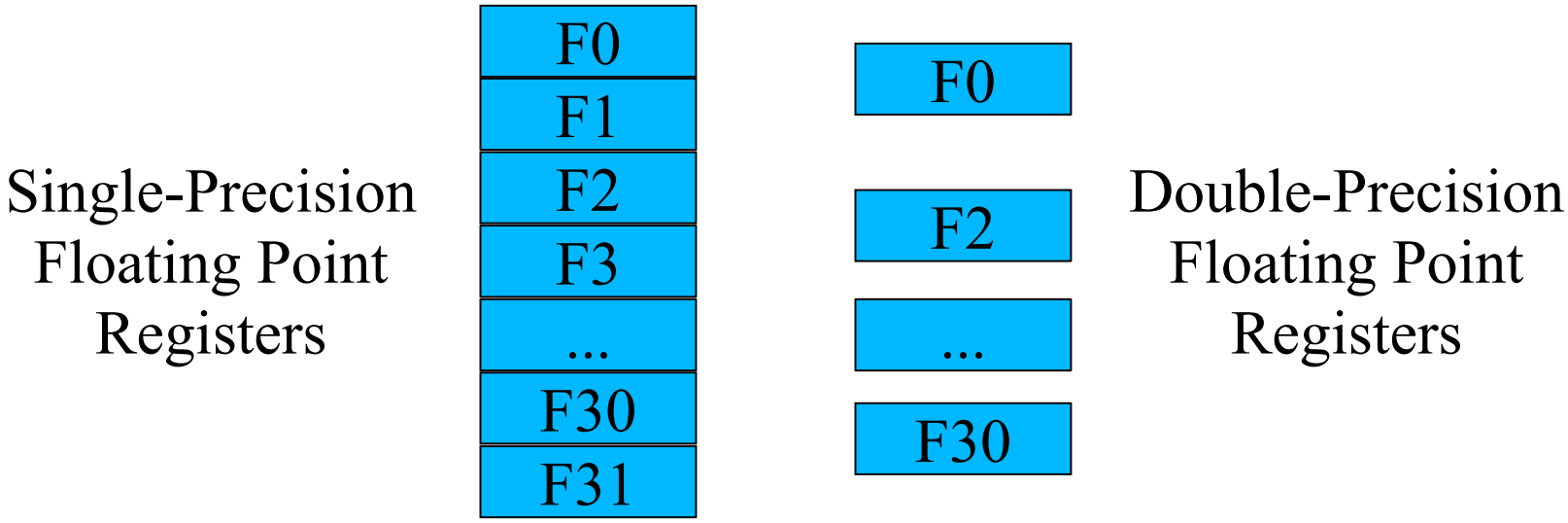
- I bit dei registri sono numerati come 0-31, da sinistra a destra (0 è il bit più significativo, 31 è quello meno significativo).
- L'ordinamento dei byte è fatto in modo simile



- Un registro può essere caricato con
  - un byte (8-bit)
  - un halfword (16-bit)
  - una fullword (32-bit)

Registri Floating-Point

- 32 registri da 32 bit a singola precisione (F0, F1, .., F31)
- Condivisi con16 registri da 64 bit a doppia precisione (F0, F2, ..., F30)
- La più piccola unità indirizzabile è in un FPR è 32 bit



Registri speciali

- Ci sono tre registri speciali
  - ***PC***, Program Counter, contiene l'indirizzo dell'istruzione dal leggere dalla memoria (32 bit)
  - ***IAR***, Interrupt Address Register, mantiene l'indirizzo di ritorno di 32 bit del programma interrottoquando una istruzione *TRAP* viene eseguita (32 bit)
  - ***FPSR***, Floating-Point Status Register, utilizzato nei conditional branch per valutare il risultato di una operazione FP (1 bit)

Classi di Istruzioni

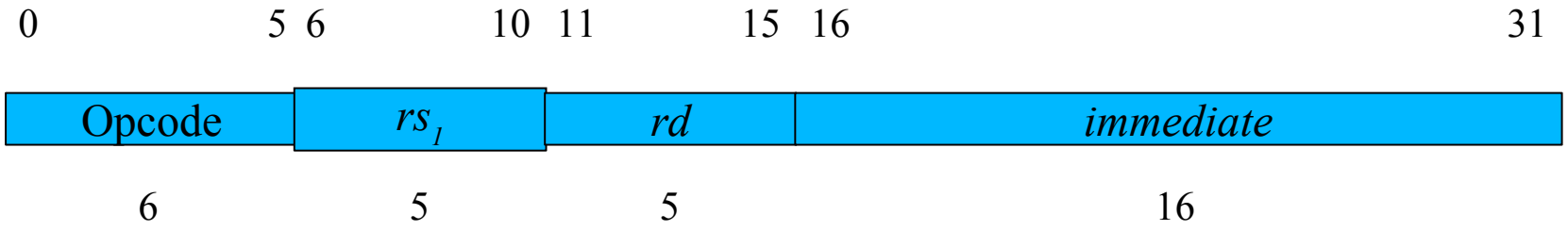
- Il DLX contiene 92 istruzioni divise in 6 classi
  - load & store instructions
  - move instructions
  - arithmetic and logical instructions
  - floating-point instructions
  - jump & branch instructions
  - special instructions

Tipi di istruzioni

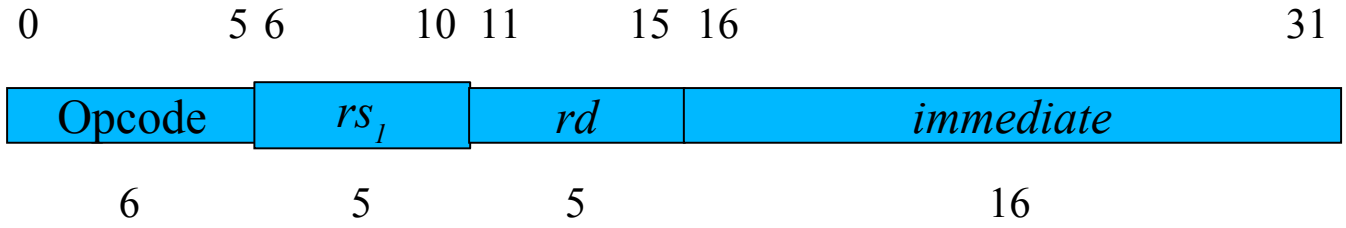
- Tutte le istruzioni DLX sono di 32 bit e devono essere allineate in memoria a word.
- Esistono 3 formati delle istruzioni:
  - I-type (Immediate): manipolano dati forniti da un campo di 16 bit;
  - R-type (Register): manipolano dati forniti da uno o due registri;
  - J-type (Jump): per specificare l'indirizzo di salto non un registro;

I-type Instructions (1 of 3)

- load/store (word, u/s halfword, u/s byte)
- operazioni ALU con operando immediato
- tutte le istruzioni di salto condizionato (branch)
- JR, JALR

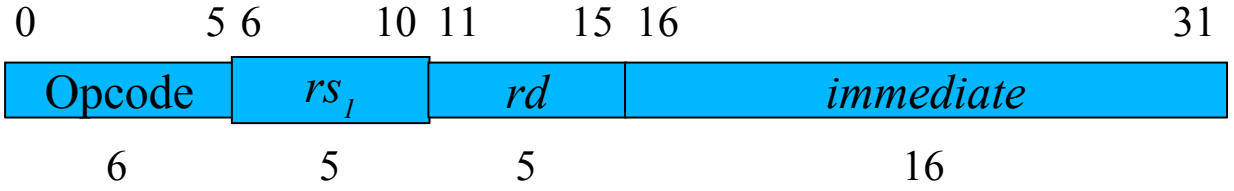


I-type Instructions (2 of 3)



- ***Opcode***: istruzione DLX da eseguire
- ***rs*$_1$**: sorgente per l'ALU, indirizzo base per le Load/Store, registro da testare per i conditional branches, indirizzo destinazione per JR & JALR
- ***rd***: destinazione per la Load e le operazioni ALU operations, sorgente per la store. (Non usato per conditional branches, JR e JALR)
- ***immediate***: spiazzamento per calcolare l'indirizzo delle load e delle store, operando per l'ALU, spiazzamento esteso in segno da sommare al PC per calcolare l'indirizzo destinazione di un conditional branch. (non usato per JR e JALR)

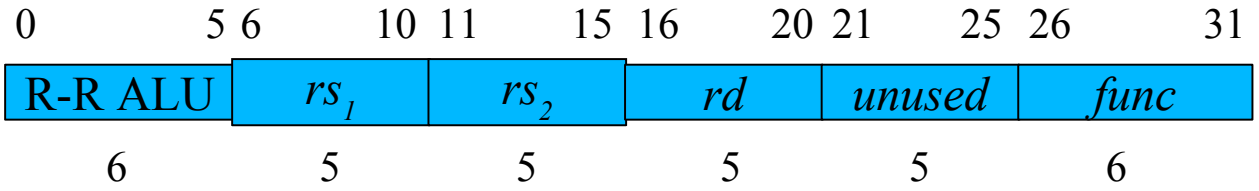
I-type Instructions (3 of 3)



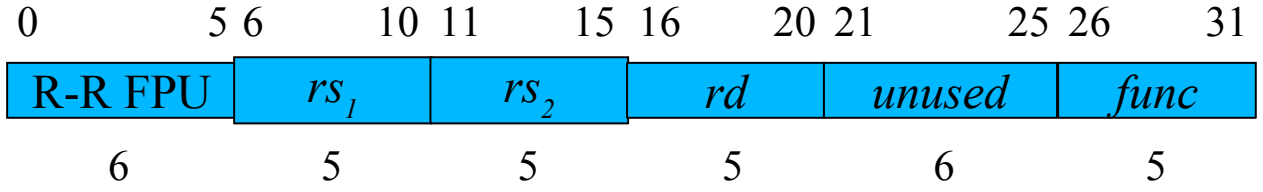

```
lw r3, 6(r2)        ; rd=r3, rs1=r2, imm=6 ;
     r3=Mem[est_segno(6)+r2];

sw -7(r4),r3        ; rd=r3, rs1=r4, imm=7 ;
     Mem[est_segno(-7)+r4]=r3;

addi r1,r2,5        ; rd=r1, rs1=r2, imm=5 ;
     r1=r2+est_segno(5);

beqz r1,target      ; rs1=r1, imm=target    ;
     if(r1==0) PC=PC+est_segno(target)

jr   r1             ; rs1=r1        ;
     PC=r1
```

R-type Instructions

- Sono usate per istruzioni che hanno come operandi dell'ALU solo registri, per leggere e scrivere su e da registri speciali (*IAR* e *FPSR*), e per spostare valori tra i GPR e/o i FPR



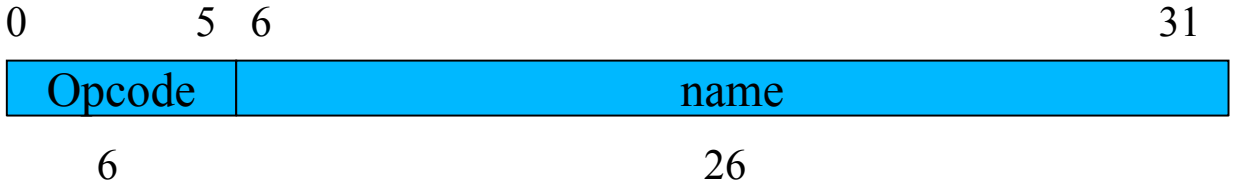

```
add   r1,r2,r3 ;rd=r1, rs1=r2, rs2=r3
```




```
addf f1,f2,f3 ;rd=f1, rs1=f2, rs2=f3
```

J-type Instructions

- Include jump (J), jump & link (JAL), TRAP e return from exception (RFE)



- *name*: spiazzamento con segno di 26 bit che è sommato all'indirizzo (PC+4) per generare l'indirizzo di destinazione. Per le TRAP esso specifica un indirizzo assoluto senza segno di 26 bit.

```
j   target;
PC=PC+est_segno(target)
```