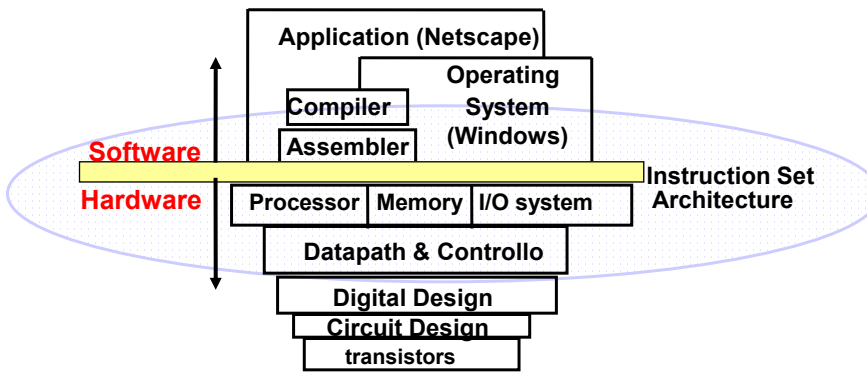
Architettura del set di istruzioni (ISA)

Instruction Set Architecture



Design Space of ISA

Five Primary Dimensions

• Number of explicit operands	(0, 1, 2, 3)
• Operand Storage	Where besides memory?
• Effective Address	How is memory location specified?
• Type & Size of Operands	byte, int, float, vector, . . .
	How is it specified?
• Operations add, sub, mul, . . .	How is it specifed?

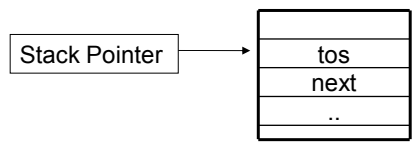
Other Aspects

• Successor	How is it specified?
• Conditions	How are they determined?
• Encodings	Fixed or variable? Wide?
• Parallelism	

Classificazione dell'ISA: Stack

Gli operandi sono implicitamente sulla cima dello stack

0 operandi `add` tos ← tos + next



E' richiesto che preventivamente gli operandi siano caricati dalla memoria mediante operazioni di push

I risultati vengono copiati dalla cima dello stack in memoria mediante una pop

Classificazione dell'ISA: accumulatore

Un operando è implicitamente l'accumulatore

1 operando	`add A`	acc ← acc + mem[A]
1+x operandi	`addx A`	acc ← acc + mem[A + x]

- E' necessario caricare il valore nell'accumulatore dalla memoria mediante una load

Classificazione dell'ISA

- **Memoria-Memoria:**
  - 2 operandi `add A B` mem[A] ← mem[A] + mem[B]
  - 3 operandi `add A B C` mem[A] ← mem[B] + mem[C]
- **Registro Memoria:**
  - 2 operandi `add Ra,C` Ra ← Ra+mem[C]
  - 3 operandi `add Ra,Rb,C` Ra ← Rb+mem[C]
- **Registro-Registro (Load/Store):**
  - 3 operandi `add Ra Rb Rc` Ra ← Rb + Rc

Classificazione dell'ISA

Number of memory address	Maximum number of operands allowed	Examples
0	3	SPARC, MIPS, PowerPC, ALPHA
1	2	Intel 80x86, Motorola 68000
2	2	VAX (also has three-operans formats)
3	3	VAX (also has three-operans formats)

Confronto

- In riferimento all'istruzione di alto livello A:= B+C

STACK	Accumulatore	Registro-Registro	Memoria-Registro	Memoria-Memoria
push B push C add Pop A	load B Add c Store a	Load r1,B Load r2,C Add r3,r1,r2 Store A, r3	Load r1, B Add r2, r1, C Store A, r2	Add A,B,C

Alcuni esempi

Machine	Number of registers	Architecture Style	Year
EDSAC	1	accumulator	1949
IBM 701	1	accumulator	1953
CDC 6600	8	load-store	1963
IBM 360	16	register-memory	1964
DEC PDP-8	1	accumulator	1965
DEC PDP-11	8	register-memory	1970
Intel 8008	1	accumulator	1972
Motorola 6800	2	accumulator	1974
DEC VAX	16	register-memory, memory-memory	1977
Intel 8086	1	extended accumulator	1978
Motorola 68000	16	register-memory	1980
Intel 80386	8	register-memory	1985
MIPS	32	load-store	1985
HP PA-RISC	32	load-store	1986
SPARC	32	load-store	1987
PowerPC	32	load-store	1992
DEC Alpha	32	load-store	1992

General Purpose Registers

- 1975-2011 tutte le macchine usano general purpose registers
- Vantaggi dei registri:
  - Sono più veloci della memoria
  - Sono più facili da usare per un compilatore
    - Es., (A*B) – (C*D) – (E*F) A differenza dello stack, i prodotti possono essere eseguiti in qualsiasi ordine
  - Possono conservare variabili
    - Il traffico con la memoria è ridotto, cosicchè i programmi sono più veloci
  - La densità del codice aumenta (poichè la codifica del nome di un registro richiede meno bit dell'indirizzo di una locazione di memoria)

Indirizzamento della memoria

- Ogni macchina usa indirizzi a livello di 8 bit (byte)
- Due problemi per il progetto dell' ISA:
  - Dal momento che una word di 32 bit può essere letta come una sequenza di quattro byte, come mappare gli indirizzi a livello di byte all'interno della word ? (big endian vs little endian)
  - L'indirizzo iniziale di una word può essere qualsiasi?

Ordinamento degli indirizzi

- Big Endian:

  L'indirizzo del byte più significativo = indirizzo word (xx00 = Big End of word)
  - IBM 360/370, Motorola 68k, MIPS, Sparc, HP PA
- Little Endian:

  L'indirizzo del byte meno significativo = indirizzo word (xx00 = Little End of word)
  - Intel 80x86, DEC Vax, DEC Alpha (Windows NT)

3	2	1	0
7	6	5	4

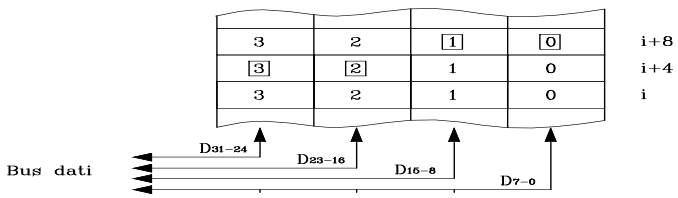
Little Endian

0	1	2	3
4	5	6	7

Big Endian

Allineamento in memoria

- L'indirizzo può essere
  - allineato (solo indirizzi multipli della dimensione della word)



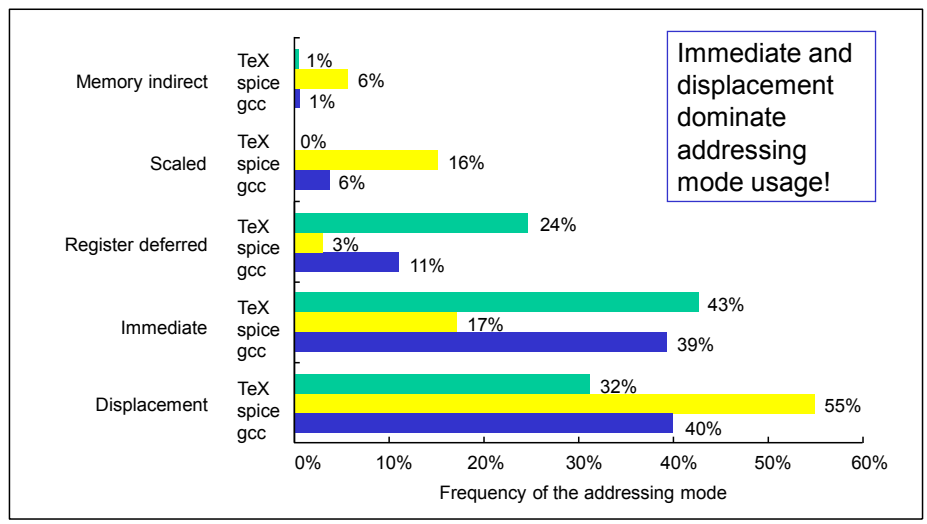
Modalità di indirizzamento

Addressing mode	Example instruction	Meaning	When used
Register	`Add R4,R3`	`Regs[R4]←Regs[R4]+ Regs[R3]`	*When a value is in a register.*
Immediate	`Add R4,#3`	`Regs[R4]←Regs[R4]+3`	*For constants.*
Displacement	`Add R4,100(R1)`	`Regs[R4]←Regs[R4]+ Mem[100+Regs[R1]]`	*Accessing local variables.*
Register deferred or indirect	`Add R4,(R1)`	`Regs[R4]←Regs[R4]+ Mem[Regs[R1]]`	*Accessing using a pointer or a computed address.*
Indexed	`Add R3,(R1 + R2)`	`Regs[R3]←Regs[R3]+ Mem[Regs[R1]+Regs[R2]]`	Sometimes useful in array addressing: R1 = *base of array*; R2 = *index amount.*
Direct or absolute	`Add R1,(1001)`	`Regs[R1]←Regs[R1]+ Mem[1001]`	Sometimes useful for accessing static data; address constant may need to be large.

Modalità di indirizzamento

Memory indirect or memory deferred	`Add R1,@(R3)`	`Regs[R1]←Regs[R1]+ Mem[Mem[Regs[R3]]]`	*If* R3 *is the address of a pointer* p*, then mode yields* *p.
Autoincrement	`Add R1,(R2)+`	`Regs[R1]←Regs[R1]+ Mem[Regs[R2]] Regs[R2]←Regs[R2]+d`	*Useful for stepping through arrays within a loop.* R2 *points to start of array; each reference increments* R2 *by size of an element,* d.
Auto-decrement	`Add R1,Ð(R2)`	`Regs[R2]←Regs[R2]Ðd Regs[R1]←Regs[R1]+ Mem[Regs[R2]]`	**Same use as autoincrement. Autodecrement/increment can also act as push/ pop to implement a stack.**
Scaled	`Add R1,100(R2)[R3]`	`Regs[R1]←Regs[R1]+ Mem[100+Regs[R2]+Regs[R3]*d]`	*Used to index arrays. May be applied to any indexed addressing mode in some machines.*

Mesurement on add. modes



Uso delle modalità di indirizzamento

programmi misurati su una macchina con tutte le modalità di indirizzamento (VAX)

--- Displacement: 42% avg, 32% to 55% ↕ 75%

--- Immediate: 33% avg, 17% to 43% ↕ 88%

--- Register deferred (indirect): 13% avg, 3% to 24% ↕

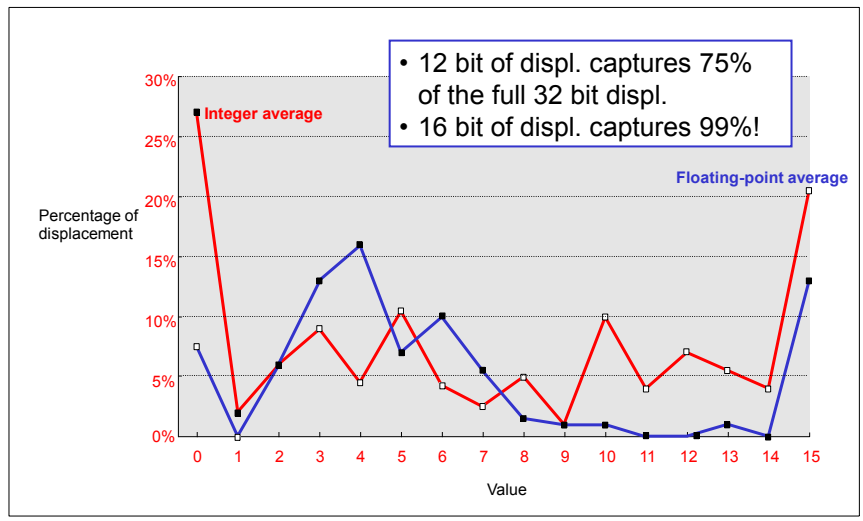
--- Scaled: 7% avg, 0% to 16%

--- Memory indirect: 3% avg, 1% to 6%

--- Misc: 2% avg, 0% to 3%

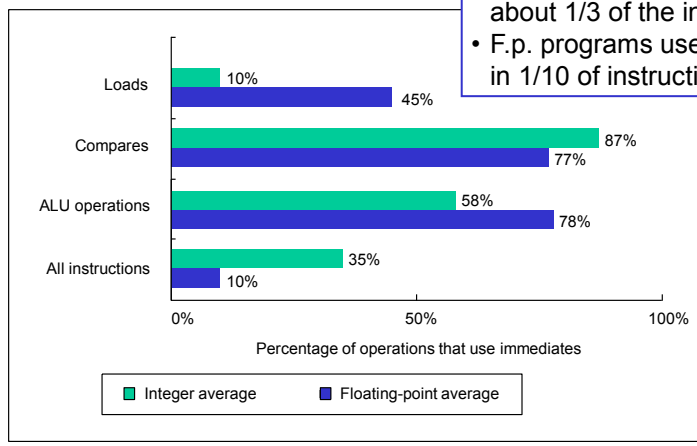
75% displacement & immediate
85% displacement, immediate & register indirect

Range of displacement



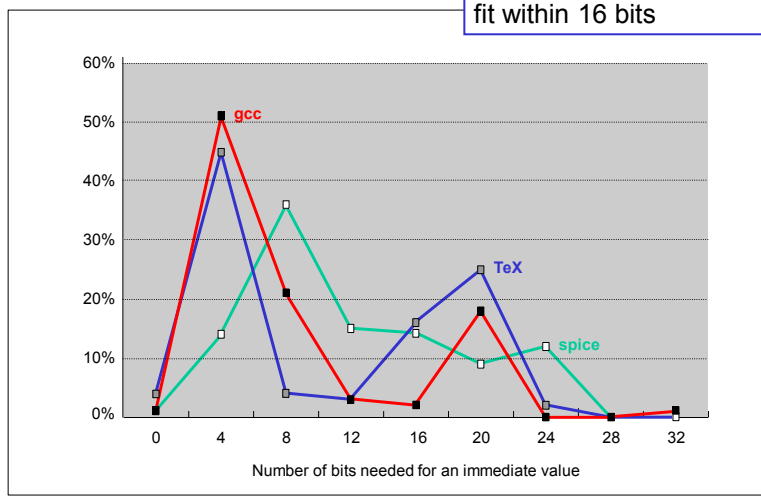
Where Immediate has to be used?

- Int programs use immediate in about 1/3 of the instructions
- F.p. programs use immediate in 1/10 of instructions!



Values for Immediate

75% to 80% of the immediate fit within 16 bits



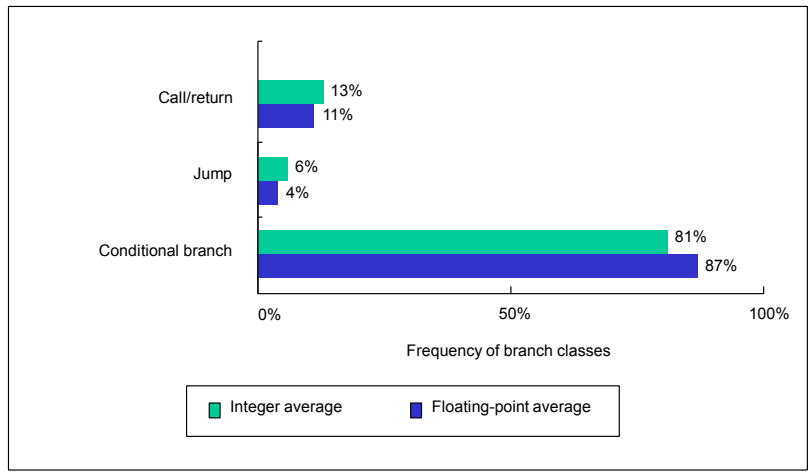
Modalità di indirizzamento (controllo)

- Salto, salto condizionato, chiamata e ritorno da sottoprogrammi
  - Diretto
  - Relativo al PC o ad altro registro

- Esempi

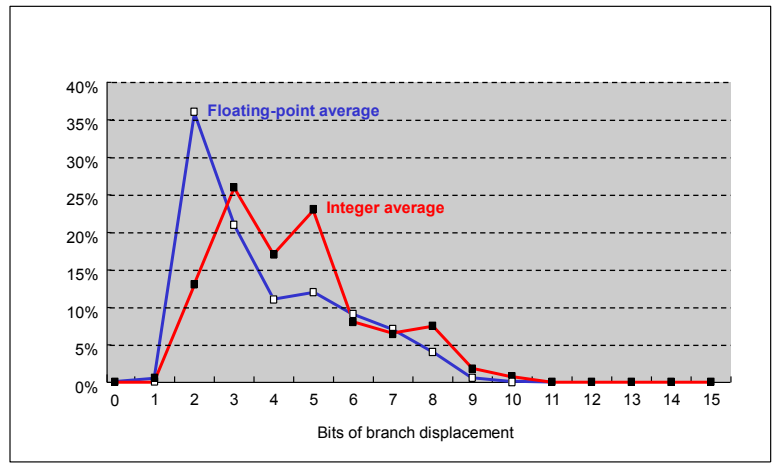
```
JMPDEST        ; Diretto o relativo a PC
JZ   wait      ; Di solito relativo a PC
call sub       ; Di solito diretto
BR   R30       ; EA destinazione = R30
```


Mesurements on Instruction for Control Flow



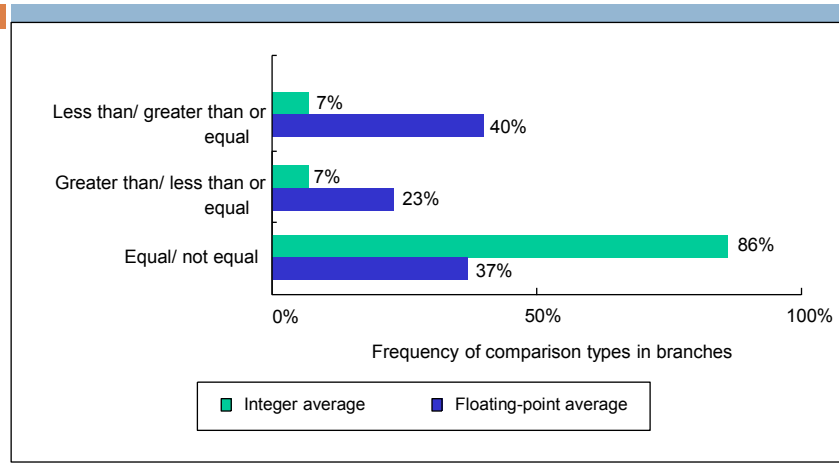
Breakdown of control flow instructions into three classes: calls or returns, jumps, and conditional branches.

Size for Branch Displacement



Branch distances in terms of number of instructions between the target and the branch instruction.

Mesurements on Conditional Branches



Frequency of different types of compares in conditional branches.

Indirizzamento: riassumendo

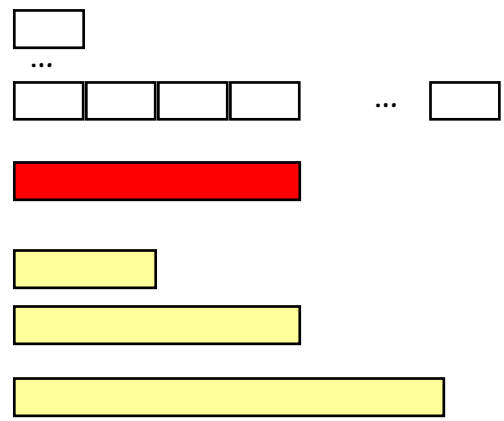
- Modalità di indirizzamento dei dati più importanti:
  - Displacement, Immediate, Register Indirect
- Dimensione dello spiazzamento:
  - dai 12 ai 16 bit
- Dimensione degli operandi immediati:
  - dagli 8 ai 16 bit

Formato delle istruzioni

Variabile:
(VAX)

Fissa:
(MIPS,PowerPC)

Ibrida:
(Intel 80x86)

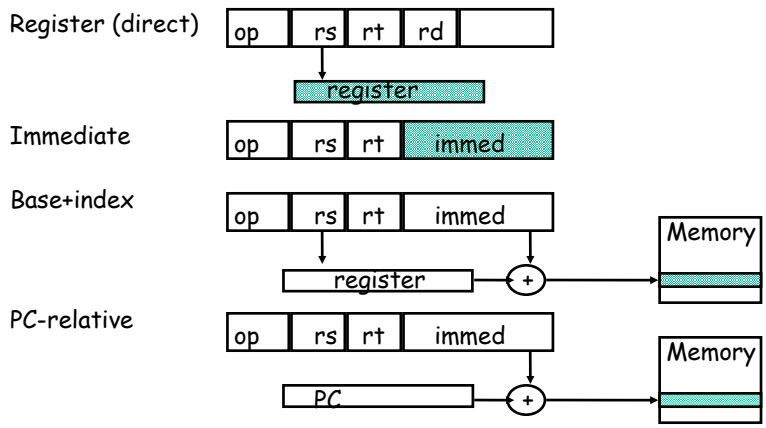


Formato delle istruzioni

- Se la dimensione del codice è il requisito più importante, si usano istruzioni a lunghezza variabile
  - Questo stile è il migliore quando sono presenti molte modalità di indirizzamento e operandi
- Se le prestazioni sono il requisito più importante, si usano istruzioni a lunghezza fissa
  - Questa scelta è particolarmente indicata quando l'operazione e la modalità di indirizzamento sono combinate nel codice operativo
  - Lavora bene quando sono presenti poche modalità di indirizzamento

MIPS: Modalità di indirizzamento
Formato Istruzioni

- Tutte le istruzioni solo di 32 bit

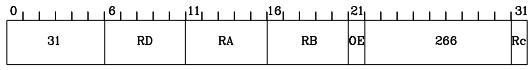


Formato Power PC

Formato istruzioni aritmetiche

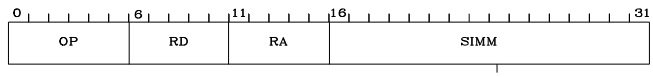
Esempio di istruzione aritmetica:
Istruzione di somma

```
ADD   RD,RA,RB     (OE=0, Rc=0)
ADD.  RD,RA,RB     (OE=0, Rc=1)
ADDO  RD,RA,RB     (OE=1, Rc=0)
ADDO. RD,RA,RB     (OE=1, Rc=1)
```

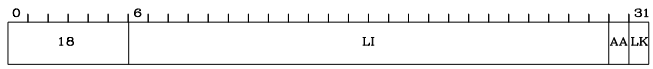


PowerPC

Formato istruzioni aritmetiche con operando immediato e istruzioni di caricamento e e memorizzazione dei registri



Formato istruzione di salto incondizionato



Formato Intel

PREFISSI				ISTRUZIONE				
Istruzione	Dimensione Indirizzo	Dimensione Operando	Segmento	Codice	MOD R/M	SIB	Scostamento	Immediato
0/1	0/1	0/1	0/1	1/2	0/1	0/1	0/1/2/4	0/1/2/4

Numero di byte possibili per ciascuno dei campi dell'istruzione

Esempio

MOV EAX, ES:V[EBX+EIX]	6 byte
PUSH EAX	1 solo byte
PUSH Var[EBX]	fino a 6 byte