Livello ISA

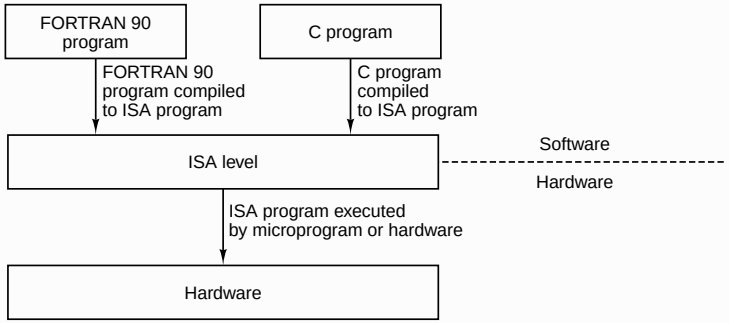
Instruction Set Architecture:
l'insieme delle istruzioni eseguibili dal processore: linguaggio macchina

Presentazione generale dei linguaggi macchina.

- Elenco degli aspetti salienti di un linguaggio macchina.
- Confronto tra linguaggi macchina.

Software – Hardware

Scelta dell'ISA importante perché definisce l'interfaccia tra hardware e software.
Tutto il codice tradotto in istruzioni macchina.
Scopo dell'hardware: eseguire istruzioni macchina.



Progettazione di un ISA

Diverse filosofie:

- CISC (Complex Instruction Set Computer) Linguaggio macchina simile ai linguaggi alto livello
- RISC (Reduced Instruction Set Computer) Istruzioni semplici e veloci.
- VLIW (Very Long Instruction Word) Sfruttare il parallelismo del processore.

Compromesso tra due esigenze:

- Progettisti hardware
- Programmatori.

Progettisti hardware

Desiderata: istruzioni

- facilmente implementabili,
- sfruttano le potenzialità della tecnologia,
- ne evitano i difetti.

Esempi:

- solo istruzioni aritmetiche-logiche (nessuna funzione analitica),
- molti registri generici e pochi accessi alle memoria,
- istruzioni vettoriali (SSE), VLIW,
- istruzioni di salto posticipato, con suggerimento; istruzioni condizionate.

Compilatori, programmatori

Desiderata:

- le istruzioni fondamentali e quelle utili devono essere presenti,
- istruzioni con comportamento chiaro e prevedibile.

Inoltre: le istruzioni più usate nei programmi vanno implementate in maniera efficiente.

Compatibilità con il software

Terzo vincolo (retro) compatibilità con il software esistente.
Esempio primicia: l'ISA IA-32 (Intel Architecture a 32 bit) (x86):
4004 – 8080 – 8086 – 80286 – 80386 (IA-32) –
MMX – 3DNow – SSE –
x86-64 (AMD64).

Eccezioni alla compatibilità

- Apple: Motorola 68000 — PowerPc — IA-32 (x86);
- Sun: Motorola 68000 — Sparc;
- Intel: Pentium-Core (IA-32) — Itanium (IA-64);
- Video giochi, sistemi embedded.
  Sony Play Station – PS1, PS2: MIPS, PS3: Cell, PS4: x86.

Prevedere il futuro:

Un ISA può durare diversi decenni.

Deve adattarsi alla sviluppo tecnologico.
Nella definizione di un ISA bisogna favorire e semplificare le future estensioni.

Nel futuro è probabile un aumento dello:
- spazio di memoria indirizzabile,
- numero di registri interni,
- lunghezza della parola,
- insieme di istruzioni eseguibili.

Caratteristiche dei linguaggi macchina (ISA)

- specifica
- istruzioni
- tipi di dato
- modello memoria
- formato istruzioni
- indirizzamento
- modalità di funzionamento
- I/0

Specifica

Definire il comportamento del processore:

- formalmente: SPARC V9, ARM
  informative (descrizione intuitiva)
  normative (descrizione formale e rigorosa)
  alcune scelte lasciate all'implementazione.
- informalmente: IA32 (Pentium-Core)
  non sono state pubblicate delle specifiche formali

Differenza dovuta a motivi commerciali.

Tipi di istruzioni

Un catalogazione:

- Movimento dati: memoria - registri:
  LOAD reg add, STORE reg add, MOVE reg reg.
- Operazioni: aritmetiche, logiche, rotazioni (in diversi ISA varia il numero di argomenti):
  ADD reg reg reg, NOT reg reg, SHIFT_LEFT reg reg n,
- Operazioni di salto: condizionate e non
  GOTO label, BRANCH label, BRANCH_EQZ reg label, BRANCH
- Chiamata di procedure, al sistema operativo.
  JUMPnLINK, INVOKE, SYSCALL.

Tipi di dato

Supportati dall'hardware.

- numerici:
  - naturali: unsign (8, 16, 32, 64),
  - interi: sign (8, 16, 32, 64),
  - reali: (32, 64, 128),
  - BCD: binary code decimal format
- non numerici:
  - caratteri, booleani,
  - puntatori (indirizzi di memoria),
  - stringe, vettori (sequenze di dati),

Tipi di dato: esempi

x86:

Type	8 Bits	16 Bits	32 Bits	64 Bits
Signed integer	×	×	×	× (64-bit)
Unsigned integer	×	×	×	× (64-bit)
Binary coded decimal integer	×			
Floating point			×	×

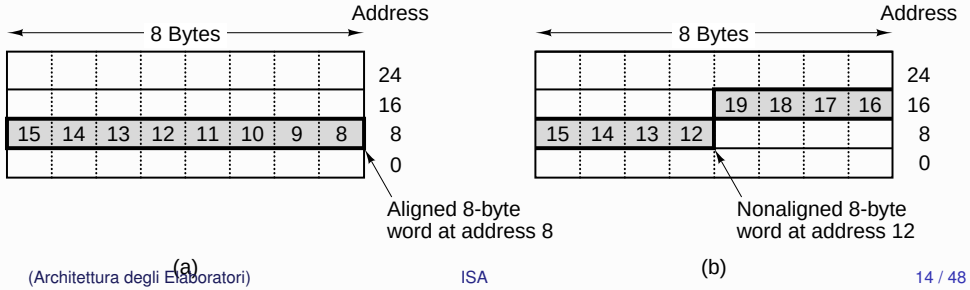
ARM:

Type	8 Bits	16 Bits	32 Bits	64 Bits
Signed integer	×	×	×	
Unsigned integer	×	×	×	
Binary coded decimal integer				
Floating point			×	×

Modello di memoria

Principali aspetti:

- Spazio di memoria: unico o diviso in istruzioni-dati.
- Lunghezza dell'unità di memoria: bit, byte, parola,
- Alcuni processori accedono solo a parole allineate.



Modello di memoria

Semantica della memoria:

- Semantica stretta: accessi in memoria nell'ordine con cui sono scritti nel codice. Semantica semplice, agevola il programmatore
- Semantica lasca: per migliore il parallelismo nel processore, sono permessi accessi "fuori ordine" alla memoria.
  Semantica complessa.

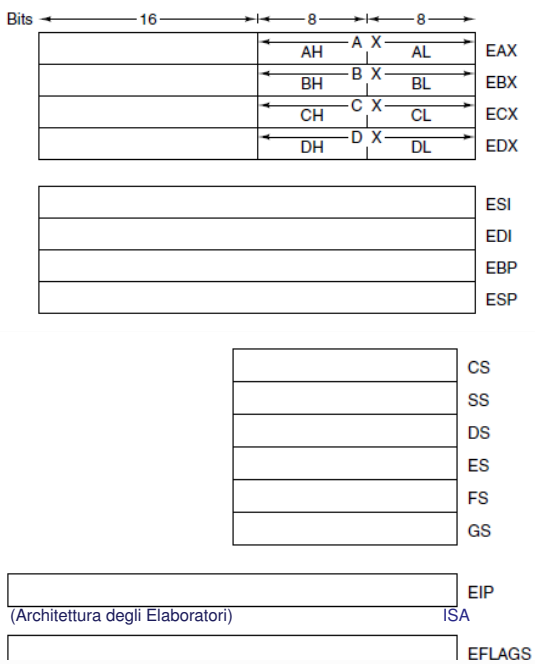
  Necessario stabilire regole e meccanismi di sincronizzazione.

Registri interni

Solo specializzati, anche generici.

- Registri generici: variabili temporanee (riduce gli accessi in memoria).
- Registri specializzati, funzioni specifiche.
  Nel IA-32 tutti i registri etichettati con un funzione specifica
  In tutti i processori troviamo:
  - PC: program counter;
  - IR: instruction register;
  - PSW: program status word,
    descrive lo stato del programma: codici di condizione, modalità macchina, livello di priorità, abilitazione degli interrupt (MIPS: Status)

Registri interni x86



Registri interni ARM

Register	Alt. name	Function
R0–R3	A1–A4	Holds parameters to the procedure being called
R4–R11	V1–V8	Holds local variables for the current procedure
R12	IP	Intraprocedure call register (for 32-bit calls)
R13	SP	Stack pointer
R14	LR	Link register (return address for current function)
R15	PC	Program counter

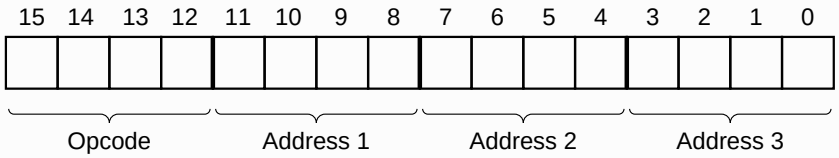
Formato delle istruzioni

Come le istruzioni sono rappresentate nel calcolatore.

Un'istruzione assembly:
`ADDS r2 r3 #4`
viene codificata, in linguaggio macchina da una sequenza di bit che deve indicare:

- il tipo di istruzione `ADD`;
- gli argomenti su cui opera: i registri `r2 r3`. la constante `4`;
- eventuali altre indicazioni `S`

Formato delle istruzioni

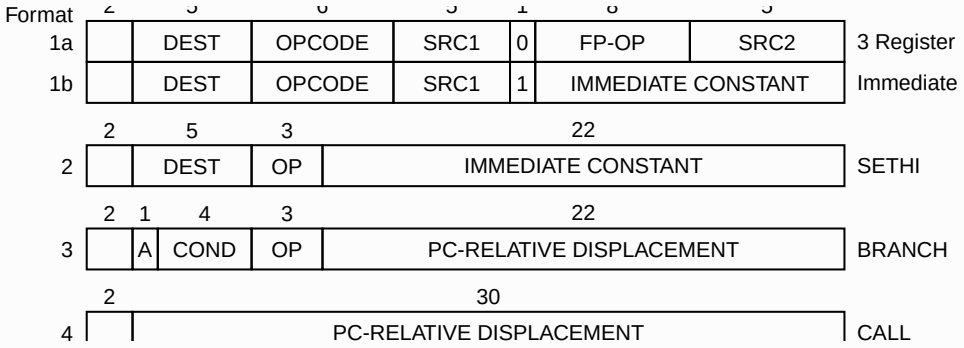


Obiettivi:

- Semplificare la decodifica da parte del processore;
- ridurre la lunghezza delle istruzioni;
- facilitare l'estensione dell'ISA, l'aggiunta di nuove istruzioni.

Esempio: Sparc

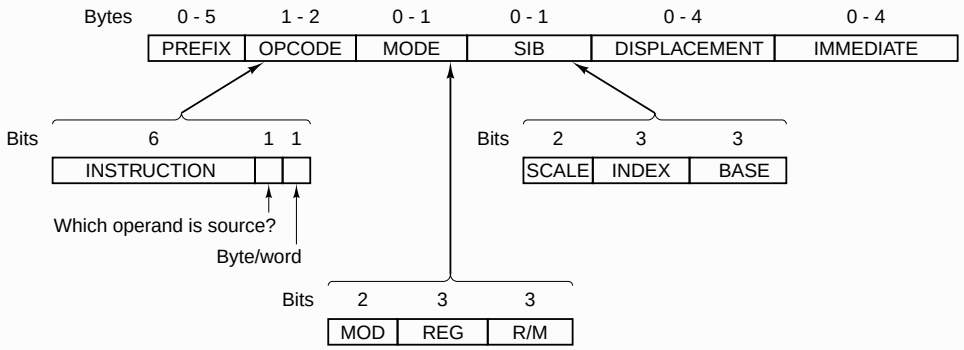
Formati diversi a seconda del numero argomenti funzione.



Esempio: ARM

31 28	27	26	25	24	23	22	21	20	19 16	15 12	11 8	7	6	5	4	3 0	Instruction type
Cond	0	0	I	Opcode				S	Rn	Rd	Operand2						Data processing / PSR Transfer
Cond	0	0	0	0	0	0	A	S	Rd	Rn	RS	1	0	0	1	Rm	Multiply
Cond	0	0	0	0	1	U	A	S	RdHi	RdLo	RS	1	0	0	1	Rm	Long Multiply
Cond	0	0	0	1	0	B	0	0	Rn	Rd	0 0 0 0	1	0	0	1	Rm	Swap
Cond	0	1	I	P	U	B	W	L	Rn	Rd	Offset						Load/Store Byte/Word
Cond	1	0	0	P	U	S	W	L	Rn	Register List							Load/Store Multiple
Cond	0	0	0	P	U	1	W	L	Rn	Rd	Offset1	1	S	H	1	Offset2	Halfword transfer: Immediate offset
Cond	0	0	0	P	U	0	W	L	Rn	Rd	0 0 0 0	1	S	H	1	Rm	Halfword transfer: Register offset
Cond	1	0	1	L	Offset												Branch
Cond	0	0	0	1	0	0	1	0	1 1 1 1	1 1 1 1	1 1 1 1	0	0	0	1	Rn	Branch Exchange
Cond	1	1	0	P	U	N	W	L	Rn	CRd	CPNum	Offset					Coprocessor data transfer
Cond	1	1	1	0	Op1				CRn	CRd	CPNum	Op2	0	CRm			Coprocessor data operation
Cond	1	1	1	0	Op1			L	CRn	Rd	CPNum	Op2	1	CRm			Coprocessor register transfer
Cond	1	1	1	1	SWI Number												Software interrupt

Esempio: x86



Argomenti di una istruzione

A seconda del linguaggio macchina una stessa istruzione può avere un numero variabile di argomenti.
Esempio: la somma in:

- MIPS, architetture RISC: tre argomenti.
- Java bytecode: nessun argomento,
- IA-32: due argomenti,
- architetture con accumulatore: un argomento.

Indirizzamento

Modi per specificare gli argomenti di un istruzione:

- immediato: il valore esplicito – #5
- diretto: un indirizzo di memoria – (100)
- registro: un registro – $r1
- indiretto tramite registri: un registro che contiene un indirizzo di memoria – ($r1)
- indice: indirizzo base (costante) + displacement – 100($r1)
- base indice: somma di due registri – ($r1)($r2)

Indirizzamento

Addressing mode	Core i7	OMAP4430 ARM	ATmega168 AVR
Immediate	×	×	×
Direct	×		×
Register	×	×	×
Register indirect	×	×	×
Indexed	×	×	
Based-indexed		×	

Indirizzamento

Ortogonalità tra indirizzamento e istruzioni,

Istruzioni analoghe usano le stesse modalità di indirizzamento:

Esempio: tutte le operazioni aritmetiche specificano gli argomenti nello stesso modo.

Load-store architecture Nei processori RISC:

- operazioni aritmetiche-logiche solo con indirizzamento immediato o registri.
- Accesso ai dati in memoria solo con istruzioni load – store.

Modalità di funzionamento

Permetto meccanismi di protezione,
si evita che un programma usi dati non suoi.

- kernel : esegue qualsiasi istruzione, chiamate al sistema operativo;
- utente: istruzioni ristrette, programmi applicativi;

Passaggio controllato da una modalità all'altra, chiamate al sistema operativo:
`syscall`.

Modalità di funzionamento utile anche per retro compatibilità con processori precedenti.

Modalità di funzionamento - Intel Core

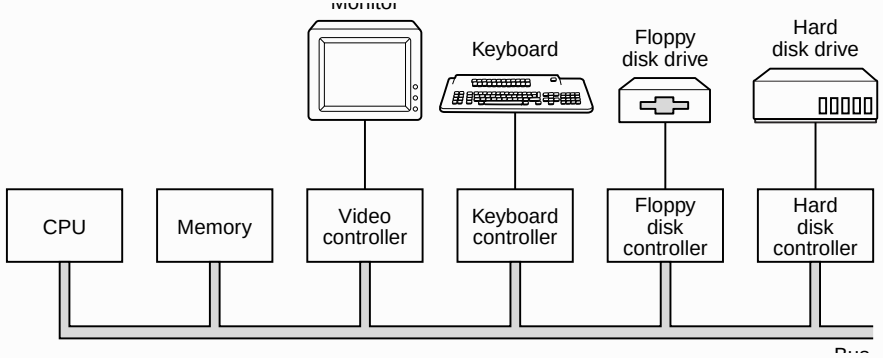
- reale: isa 8088
- virtuale: isa 8088
- protetta: isa x86-64
  - livello 0: kernel
  - livello 1: SO
  - livello 2: librerie
  - livello 3: utente

Input-Output

Sino ad ora: il funzionamento del processore (CPU) e della memoria.
Allarghiamo il quadro alle restanti parti del calcolatore: bus, memoria di massa, periferiche.

Meccanismi di comunicazione tra CPU e altri dispositivi. p. 111, 404

Calcolare: schema semplificato



Circuiti di controllo

CPU comunica con:
dispositivi di controllo, controllori: moduli di I/O, I/O channels, I/O processor (schede video, audio, di rete).

- Risiedono nel calcolatore, interfacciano CPU e periferiche.
- Trasformano:
  - comandi della CPU in segnali elettrici di comando per le periferiche,
  - i segnali dalle periferiche in dati per la CPU.
- Permetto alla CPU di interfacciarsi, in maniera uniforme, con dispositivi diversi tra loro.

Comunicazione CPU – controllori:

Attraverso dei registi (interni ai controllori).
Ogni registro è identificato da un indirizzo.
Due alternative:

- Memory mapped I/O: registri identificati da indirizzi della memoria.
  Più semplice, si usano istruzioni di accesso alla memoria (`load store`) per gestire I/O.
  Si riduce la memoria utilizzabile.
- Isolated I/0: Indirizzi e istruzioni dedicati all'I/0

La comunicazione con i controllori è simile ad accessi in memoria.

Esempio di Memory Mapped I/0

Si dive la spazio di indirizzamento in settori, ciascuno dedicato ad un diverso controllore.

Dispositivo	Address range	Size
RAM	0000 - 7FFF	32 KiB
General purpose I/O	8000 - 80FF	256 bytes
Sound controller	9000 - 90FF	256 bytes
Video controller	A000 - A7FF	2 KiB
ROM	C000 - FFFF	16 KiB

Con un semplice circuito combinatorio è possibile azionare il controllore appropriato.

Comunicazione con i controllori

Si comunica:

- dati (di ingresso o uscita)
- comandi (dalla CPU alla periferica),
- informazioni di stato sulla periferica.

Comunicazioni diverse usano registri diversi.

Esempio: una stampante.

- dati da stampare -
- comandi di stampa - interrogazione sullo stato della stampante
- informazione sull'avanzamento dei lavori, stato stampate, mancanza di carta, livello inchiostro.

Meccanismi di sincronizzazione

Necessari perché CPU e controllori funzionano a velocità differenti.
La CPU può inviare i comandi più velocemente di quanto la periferica riesca ad eseguirli

Tre meccanismi principali:

- I/O programmato con busy waiting
- I/O controllato dall'interrupt
- I/O con DMA (Direct Memory Access)

I/O programmato

La CPU controlla periodicamente lo stato di avanzamento del I/O e lo stato della periferica.
Esempio. Controllore stampante con due registri: stato (con bit READY), carattere da stampare.

- CPU - attende stampate pronta (bit READY=1);
- CPU - inserisce carattere in un registro e da comando di stampa (READY=0);
- CPU - attende, leggendo ciclicamente il bit READY, che il comando sia eseguito;
- controllore - riceve comando, lo esegue,
- controllore - segnala l'esecuzione del comando (READY=1)

I/O programmato

Meccanismo di busy waiting o polling

- semplice: non si richiede dell'hardware apposito,
- poco efficiente: spreca tempo di CPU

Usato in calcolatori semplici, sistemi embedded.

Interrupt

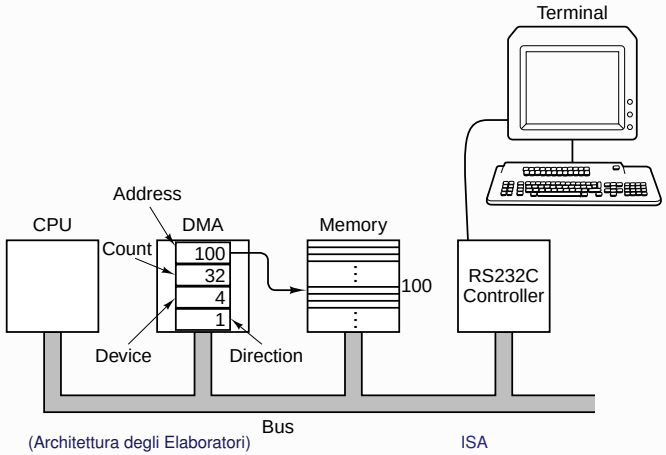
Un "campanello" (segnale di interrupt), con cui i controllori richiamano l'attenzione della CPU.

- la CPU da il comando di I/O e poi esegue altri processi,
- il controllore esegue il comando e quindi invia il segnale di interrupt
- la CPU (ricevuto l'interrupt) può decidere di sospendere il processo in esecuzione e gestire l'I/O.

Migliora notevolmente le prestazioni, ma ogni dato di I/O deve passare per la CPU
CPU distratta da continui interrupt.

DMA (Direct Memory Access)

La gestione dell'I/O viene in parte delegata ad un dispositivo (con diretto accesso alla memoria).



DMA (Direct Memory Access)

Schema di funzionamento:

- La CPU descrive al DMA l'operazione da svolgere:
  - controllore di I/O coinvolto
  - indirizzo memoria dove prelevare - depositare i dati
  - quantità di dati da trasferire
- Il dispositivo DMA gestisce il trasferimento memoria - controllore

DMA (Direct Memory Access)

- minor lavoro per la CPU, meno interrupt
- il controllore DMA come una seconda CPU: si generano conflitti per l'uso della memoria, cycle stealing
- controllore DMA può gestire una o più periferiche,
- esistono controllori DMA sofisticati (intelligenti) che eseguire codice macchina (non solo semplici comandi) (co-processore I/O).

Trap (Eccezione)

Meccanismo simile all'interrupt, forza la CPU a gestire un evento, ma segnale generato dalla CPU stessa quando incontra istruzioni anomale,

esempi: overflow, opcode inesistente, indirizzo di memoria errato, page-fault.
vantaggi: permette una gestione efficiente degli errori software, si evita di dover inserire codice di controllo.

Trap e interrupt

Meccanismi per modificare il flusso delle istruzioni, forzano la CPU a gestire un determinato evento;

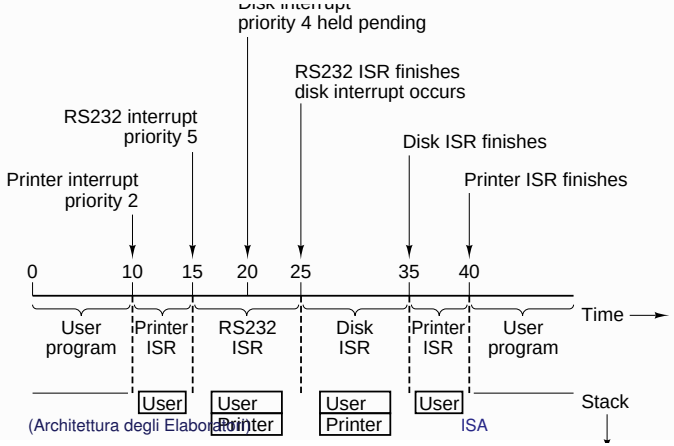
- Trap (eccezione): generato dal codice, sincrono: divisione per zero, overflow
- Interrupt: generato da una periferica, asincrono: la periferiche richiede l'attenzione della CPU

Gestione dell'interrupt (trap)

- il dispositivo genera un segnale di interrupt;
- CPU rileva il segnale, identifica il dispositivo, richiede il vettore dell'interrupt;
- con vettore dell'interupt e tabella degli interrupt determina procedure da eseguire;
- si salva PC, PSW, mascheramento di nuovi interrupt, cambio modalità;
- esecuzione della procedura (salva stato, identifica dispositivo, esamina il suo stato, . . .);
- ripristino del programma sospeso.

Mascheramento dell'interrupt

Priorità tra dispositivi: dispositivi che non posso attendere hanno priorità più alta. Esempio:



Precisione dell' interrupt:

Per la gestione dell' interrupt la CPU fermarsi su un istruzione precisa del programma.
A causa del parallelismo, nei calcolatori superscalari, questo vincolo è costoso:

- la CPU deve scegliere l'istruzione su cui interrompersi;
- completare le istruzioni precedenti;
- smettere di caricare le istruzioni successive, scaricare quelle eventualmente iniziate;

Precisione dell' interrupt:

In ISA recenti hanno proposto interrupt imprecisi: non viene definita l'istruzione precisa su cui il programma si interrompe:

- permettono implementazioni più efficienti,
- complicano la vita al programmatore.

Un altro esempio di contrapposizione tra progettazione hardware – scrittura software.