Livello "Instruction Set Architecture"

Trasferimenti

MOV DST,SRC	Trasferisce SRC in DST
PUSH SRC	Impila SRC sullo stack
POP DST	Pop di una parola dallo stack e la pone in DST
XCHG DS1, DS2	Scambia DS1 e DS2
LEA DST,SRC	Carica in DST l'indirizzo effettivo di SRC
CMOVcc DST,SRC	Trasferimento condizionato

Aritmetica

ADD DST, SRC	Somma SRC a DST
SUB DST, SRC	Sottrae SRC da DST
MUL SRC	Moltiplica EAX con SRC (senza segno)
IMUL SRC	Moltiplica EAX con SRC (con segno)
DIV SRC	Divide EDX:EAX per SRC (senza segno)
IDIV SRC	Divide EDX:EAX per SRC (con segno)
ADC DST,SRC	Somma SRC a DST e somma il bit di riporto
SBB DST,SRC	Sottrae SRC e il riporto da DST
INC DST	Incrementa DST di 1
DEC DST	Decrementa DST di 1
NEG DST	Nega DST (lo sottrae a 0)

Decimali in codifica binaria

DAA	Codifica decimale
DAS	Codifica decimale per la sottrazione
AAA	Codifica ASCII per l'addizione
AAS	Codifica ASCII per la sottrazione
AAM	Codifica ASCII per la moltiplicazione
AAD	Codifica ASCII per la divisione

Booleane

AND DST, SRC	AND di SRC e DST, risultato in DST
OR DST, SRC	OR di SRC e DST, risultato in DST
XOR DST, SRC	OR esclusivo di SRC e DST, risultato in DST
NOT DST	Rimpiazza DST con il suo complemento a uno

Scorrimento/rotazione

SAL/ SAR DST,#	Scorrimento di DST verso s/d di # bit
SHL/ SHR DST,#	Scorrimento logico di DST verso s/d di # bit
ROL/ROR DST,#	Rotazione di DST verso s/d di # bit
RCL/ RCR DST,#	Rotazione di DST attraverso il riporto di # bit

Test/confronto

TEST SRC1,SRC2	AND degli operandi, modifica EFLAGS
CMP SRC1,SRC2	Modifica EFLAGS secondo SRC1- SRC2

Trasferimento del controllo

JMP ADDR	Salta all'indirizzo ADDR
Jxx ADDR	Salti condizionati da EFLAGS
CALL ADDR	Chiama la procedura all'indirizzo ADDR
RET	Ritorno da procedura
IRET	Ritorno da interrupt
LOOPxx	Cicla finché la condizione non sia soddisfatta
INT n	Innesca un interrupt software
INTO	Innesca un interrupt se il bit di overflow = 1

Stringhe

LODS	Carica una stringa
STOS	Memorizza una stringa
MOVS	Muove una stringa
CMPS	Copia una stringa
SCAS	Scandisce una stringa

Codici di condizione

STC	Asserisce il bit di riporto in EFLAGS
CLC	Azzera il bit di riporto in EFLAGS
CMC	Complemento del bit di riporto in EFLAGS
STD	Asserisce il bit di direzione in EFLAGS
CLD	Azzera il bit di direzione in EFLAGS
STI	Asserisce il bit di interrupt in EFLAGS
CLI	Azzera il bit di interrupt in EFLAGS
PUSHFD	Impila EFLAGS sullo stack
POPFD	Fa la pop di EFLAGS dallo stack
LAHF	Carica AH da EFLAGS
SAHF	Memorizza AH in EFLAGS

Miscellanea

SWAP DST	Cambia ordinamento dei byte di DST
CWQ	Estende EAX a EDX: EAX per la divisione
CWDE	Estende il numero di 16 bit in AX a EAX
ENTER SIZE, LV	Crea record d'attivazione di SIZE byte
LEAVE	Disfa il record d'attivazione creato da ENTER
NOP	Operazione nulla
HLT	Terminazione
IN AL, PORT	Un byte in input da PORT ad AL
OUT PORT, AL	Un byte in output da AL a PORT
WAIT	Attende un interrupt

s/d = sinistra/destra
SRC = sorgente
DST = destinazione
# = numero di scorrimenti/rotazioni
LV = puntatore alle variabili locali

Figura 5.33 Alcune istruzioni intere del Core i7.

Prof. Ivan Lanese

Livello ISA

- Il livello ISA (Instruction Set Architecture) rappresenta l'interfaccia fra l'hardware ed il software
- E' costituito dall'insieme di istruzioni che il processore è in grado di eseguire (vedi il nostro linguaggio macchina Hack)
- Solitamente i compilatori traducono in istruzioni del livello ISA i programmi scritti in linguaggi di programmazione ad alto livello (vedi il compilatore gcc usato nel progetto)

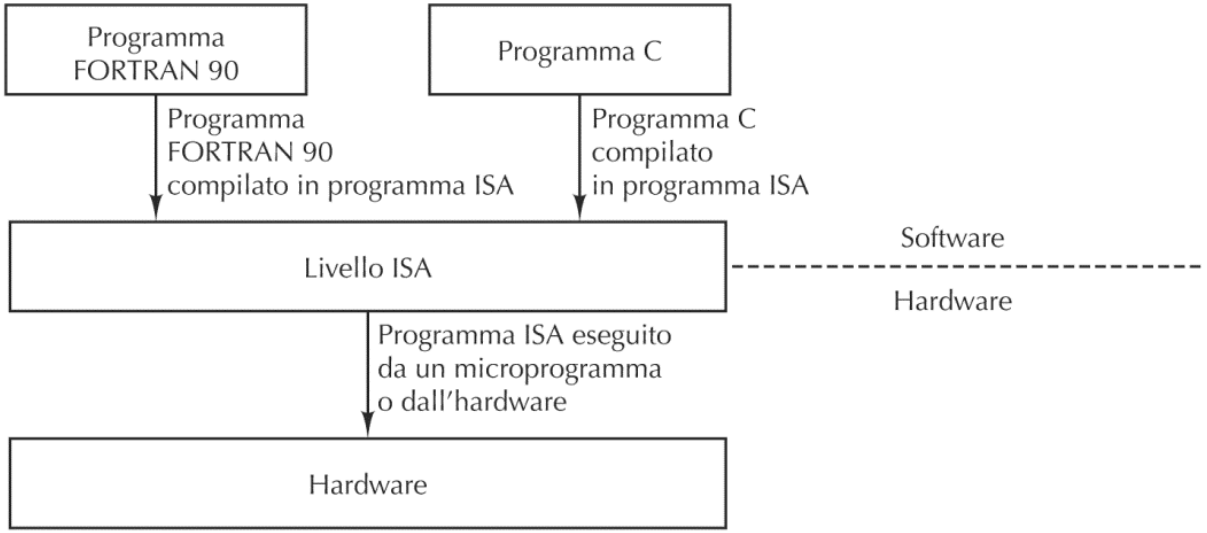

Figura 5.1 Il livello ISA è l'interfaccia tra compilatori e hardware.

Formati di istruzione

- Noi abbiamo già studiato il formato delle istruzioni per l'ISA del processore Hack
  - Lunghezza 16 bit
  - Due tipologie A-instruction (formato semplice) e C-instruction (formato "strutturato" per faciliare l'implementazione delle istruzioni al livello logico digitale, ad esempio, per controllare la ALU direttamente con i bit dell'istruzione)
- In generale le istruzioni contengono
  - Codice operativo (opcode)
    - Parte che indica il tipo di istruzione da eseguire
  - Indirizzi
    - Indicano gli operandi da usare

Formati di istruzione

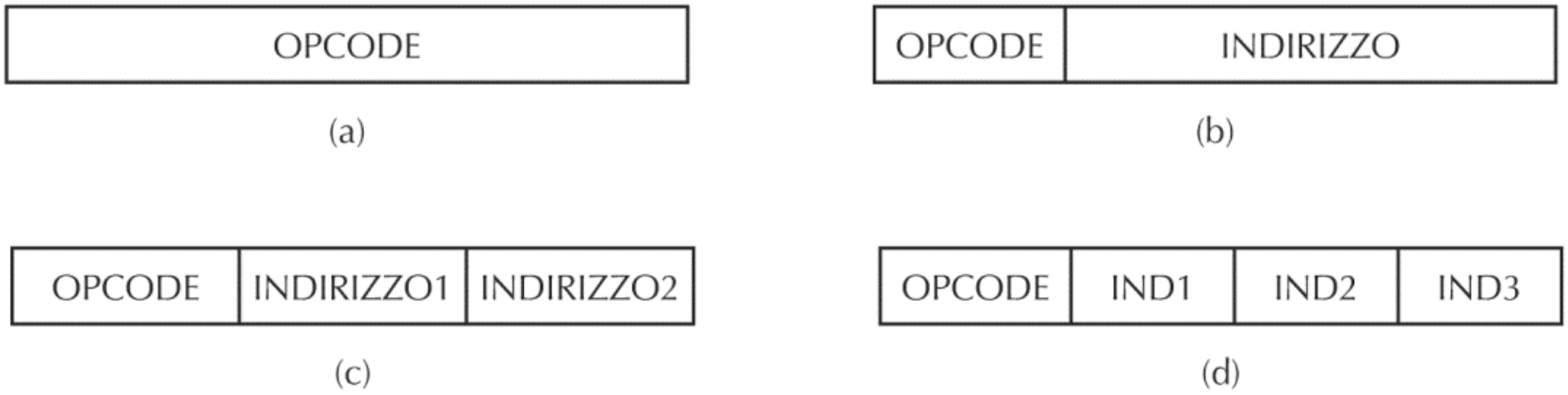

Figura 5.9 Quattro formati d'istruzioni: (a) Istruzione senza indirizzi. (b) Istruzione a un indirizzo. (c) Istruzione a due indirizzi. (d) Istruzione a tre indirizzi.

- Notate:
  - Istruzioni diverse potrebbero dedicare più o meno bit alla parte dedicata al codice operativo
  - Istruzioni diverse potrebbero avere un numero diverso di indirizzi

Modalità di indirizzamento

- La parte "indirizzi" indica dove reperire gli operandi da usare
- Esistono diversi modi per indicare come reperire tali operandi
  - Indirizzamento immediato:
    - L'istruzione include già l'operando da usare (vedi A-instruction del linguaggio Hack)
  - Indirizzamento diretto:
    - L'istruzione contiene l'indirizzo completo in memoria della cella in cui reperire l'operando
  - Indirizzamento a registro:
    - L'operando viene prelevato da un registro
  - Indirizzamento a registro indiretto:
    - L'operando viene prelevato dalla memoria, all'indirizzo puntato da un registro (vedi l'accesso in memoria del linguaggio Hack, con "indirizzamento a registro indiretto" tramite registro A)

Modalità di indirizzamento (continua)

- Indirizzamento indicizzato:
  - L'istruzione include uno "spiazzamento" (offset) che viene sommato al contenuto di un registro per ottenere l'indirizzo di memoria da cui prelevare l'operando
  - Ne riparleremo quando discuteremo la gestione della memoria nei casi di invocazione di procedure (in particolare, gestione degli "activation record")
- Indirizzamento a stack:
  - Si considera una parte di memoria da gestire secondo un modalità LIFO (last in-first out)
  - Si usa un registro interno dedicato, solitamente chiamato "Stack Pointer"
  - Le operazioni di lettura/scrittura vengono effettuate sulle celle puntate dallo stack pointer tramite operazioni push / pop
  - Anche questa modalità è usata nella gestione degli "activation record"

Tipiche istruzioni

- Grazie allo studio dell'ISA del processore Hack, abbiamo già avuto modo di vedere le tipiche istruzioni che vengono messe a disposizione:
  - Operazioni di trasferimento dati:
    - È possibile trasferire dati da memoria a registri, da registri a memoria, e da registri a registri
  - Operazioni aritmetico-logiche binarie:
    - Noi abbiamo visto solo operazioni su interi, ma solitamente esistono anche istruzioni per numeri floating-point
  - Operazioni unarie:
    - A volte sono presenti istruzioni di "shift" o rotazione (per velocizzare moltiplicazioni o divisioni per potenze di 2)
    - Operazioni molto frequenti possono avere una propria istruzione, vedi INC (incremento) caso particolare di somma
  - Salti (condizionati e incondizionati)

Invocazione di procedura

- Solitamente esistono istruzioni per l'invocazione di procedura (a differenza del semplice ISA del processore Hack che non contiene tali istruzioni)
- Esistono almeno:
  - Una istruzione per invocare una procedura
    - Si interrompe l'esecuzione sequenziale, si salta ad una nuova parte di istruzioni, ma si tiene traccia del punto da dove si è effettuata l'invocazione
  - Una istruzione di ritorno
    - Si torna ad eseguire dalla istruzione successiva a quella di invocazione

Implementazione di Invocazione di procedura

- Implementare le invocazioni di procedura richiede di allocare in memoria le informazioni necessarie per eseguire le procedure invocate
  - Tale spazio in memoria viene chiamato "record di attivazione" (activation record)
  - I record di attivazione sono organizzati a stack
    - Quando termina l'esecuzione di una procedura, si riattiva la precedente (cioè la penultima invocata)
- Vediamo un esempio di implementazione a stack delle chiamate a procedura, tramite un programma ricorsivo che risolve il problema della Torre di Hanoi

Torre di Hanoi

- Nel problema della torre di Hanoi è necessario spostare dei dischi da "Piolo 1" a "Piolo 3"
  - Spostando un disco alla volta
  - Mettendo i dischi sopra a dischi più grandi

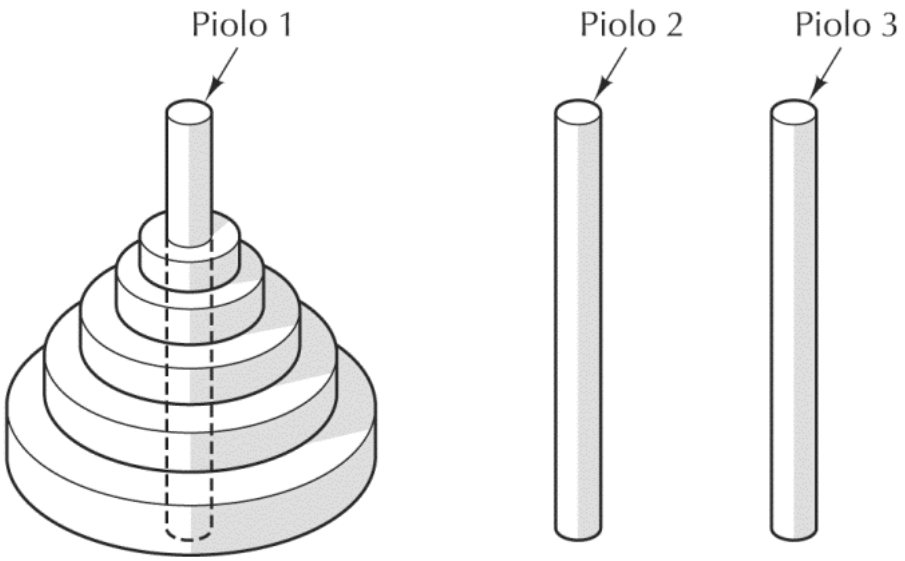

Figura 5.37 Configurazione iniziale del problema della torre di Hanoi con cinque dischi.

Torre di Hanoi

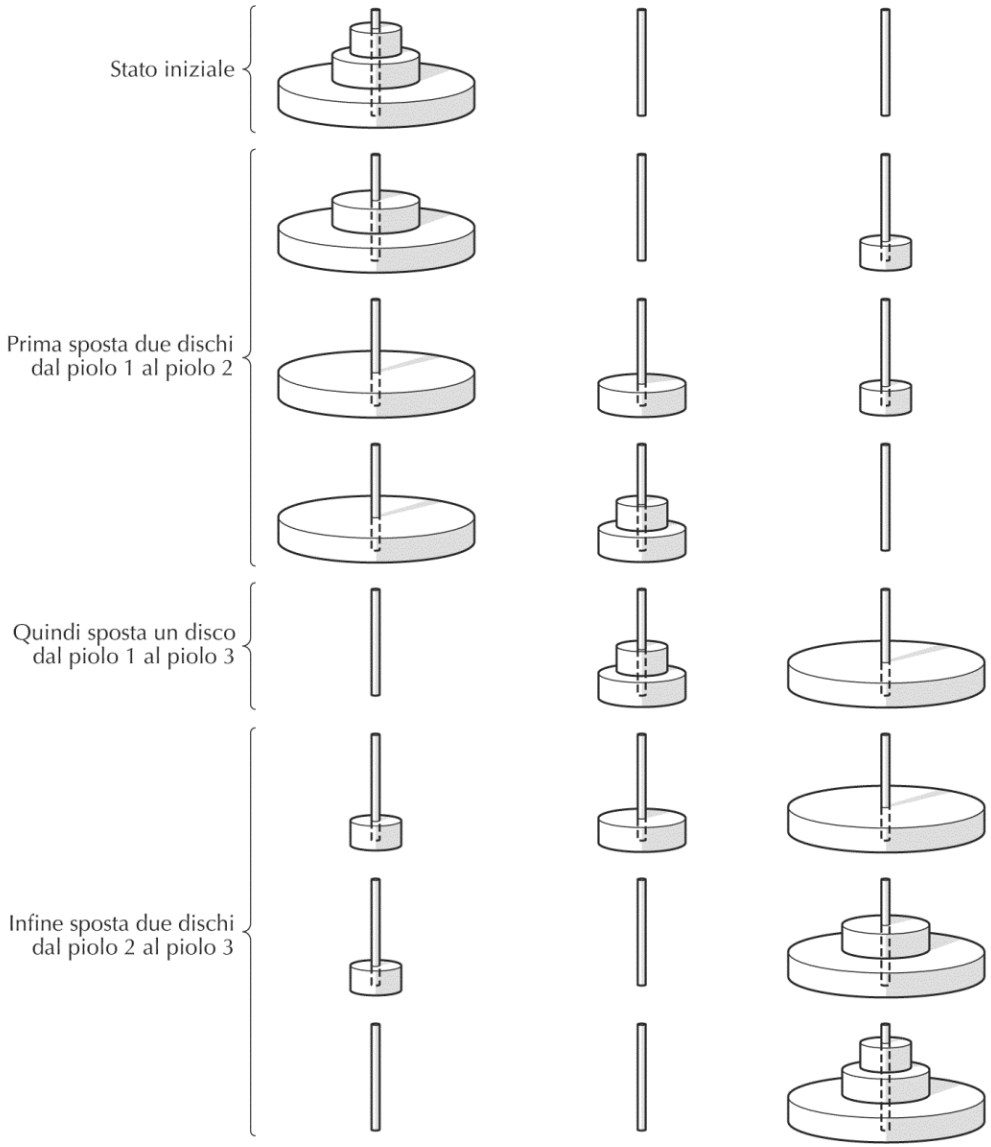

Figura 5.38 Passi necessari per la soluzione del problema della torre di Hanoi con tre dischi.

Torre di Hanoi (soluzione generale in Java)

```
public void towers(int n, int i, int j) {
    int k;

    if (n == 1)
        System.out.println( "Sposta un disco da " + i + " a " + j);
    else {
        k = 6 – i – j;
        towers(n – 1, i, k);
        towers(1, i, j);
        towers(n – 1, k, j);
  }
}
```

Figura 5.39 Procedura per la soluzione del problema della torre di Hanoi.

Torre di Hanoi (esecuzione dal punto di vista dello stack)

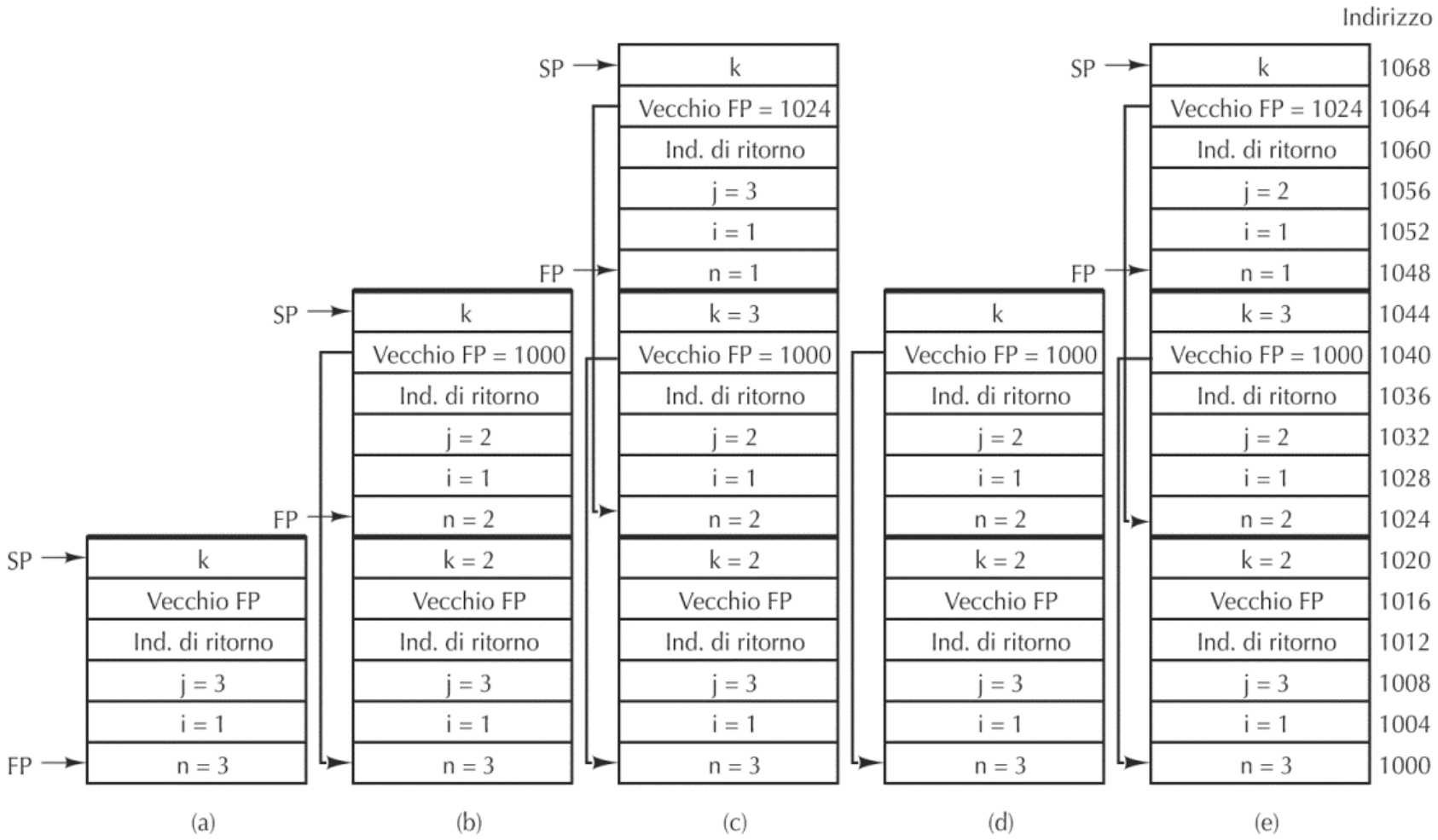

Figura 5.40 Lo stack in alcuni momenti dell'esecuzione del codice della Figura 5.39.

Record di attivazione

- Esiste solitamente un registro specializzato chiamato Frame Pointer
  - Indica il record di attivazione della funzione attualmente in esecuzione
- Nel record di attivazione vengono inseriti:
  - I parametri di invocazione
  - L'indirizzo di ritorno
    - Punto di rientro per rimettere in esecuzione il chiamante
  - Il vecchio frame pointer
    - Serve per ripristinare il Frame Pointer al momento del ritorno al chiamante
  - Variabili locali
    - Nell'esempio la variabile k

Trap

- Una trap è una chiamata di procedura automatica effettuata quando si verificano condizioni rilevanti, ad esempio
  - Opcode non definito
  - Accesso ad area di memoria non consentita
- Se si verifica uno di questi eventi, il controllo è trasferito ad una procedura detta "gestore di trap"
  - Ad esempio, il gestore di trap potrebbe comunicare l'errore e non restituire più il controllo al programma che ha generato l'errore

Interrupt

- Similmente alle "trap" le "interruzioni" (interrupt) interrompono il programma in esecuzione e trasferiscono il controllo ad un gestore deputato (detto Interrupt Service Routine – ISR)
- Mentre le trap sono "sincrone", gli interrupt sono "asincroni"
  - In questo caso "sincrono" vuol dire che la trap viene scatenata da un'istruzione errata del programma in esecuzione
  - Gli interrupt sono "asincroni" in quanto scatenati da altri dispositivi indipendentemente dal programma in esecuzione
    - Esempio: un dispositivo di I/O indica la fine di una specifica operazione
  - La tecnica dell'interrupt è fondamentale per evitare il cosiddetto "busy waiting"
    - Si parla di busy waiting quando continuamente si eseguono istruzioni per andare a verificare se è accaduto un certo evento

Interrupt: mascherabilità e priorità

- Non sempre un interrupt deve interrompere l'esecuzione attuale
  - Considerate ad esempio un gestore di un interrupt, non è detto che debba essere anch'esso interrotto
- Il sistema operativo potrebbe quindi decidere che in alcuni momenti degli interrupt non devono essere in grado di interrompere il programma in esecuzione
  - In questo caso si parla di interrupt che vengono "mascherati"
- In altri casi, conviene definire una gerarchia di interrupt, alcuni più importanti di altri
  - Un interrupt più importante può interrompere il gestore di un interrupt meno importante
  - Ma non vale il vice versa
    - In questo caso l'interrupt meno importante verrà gestito successivamente

Esempio di interrupt a diversi livelli di priorità

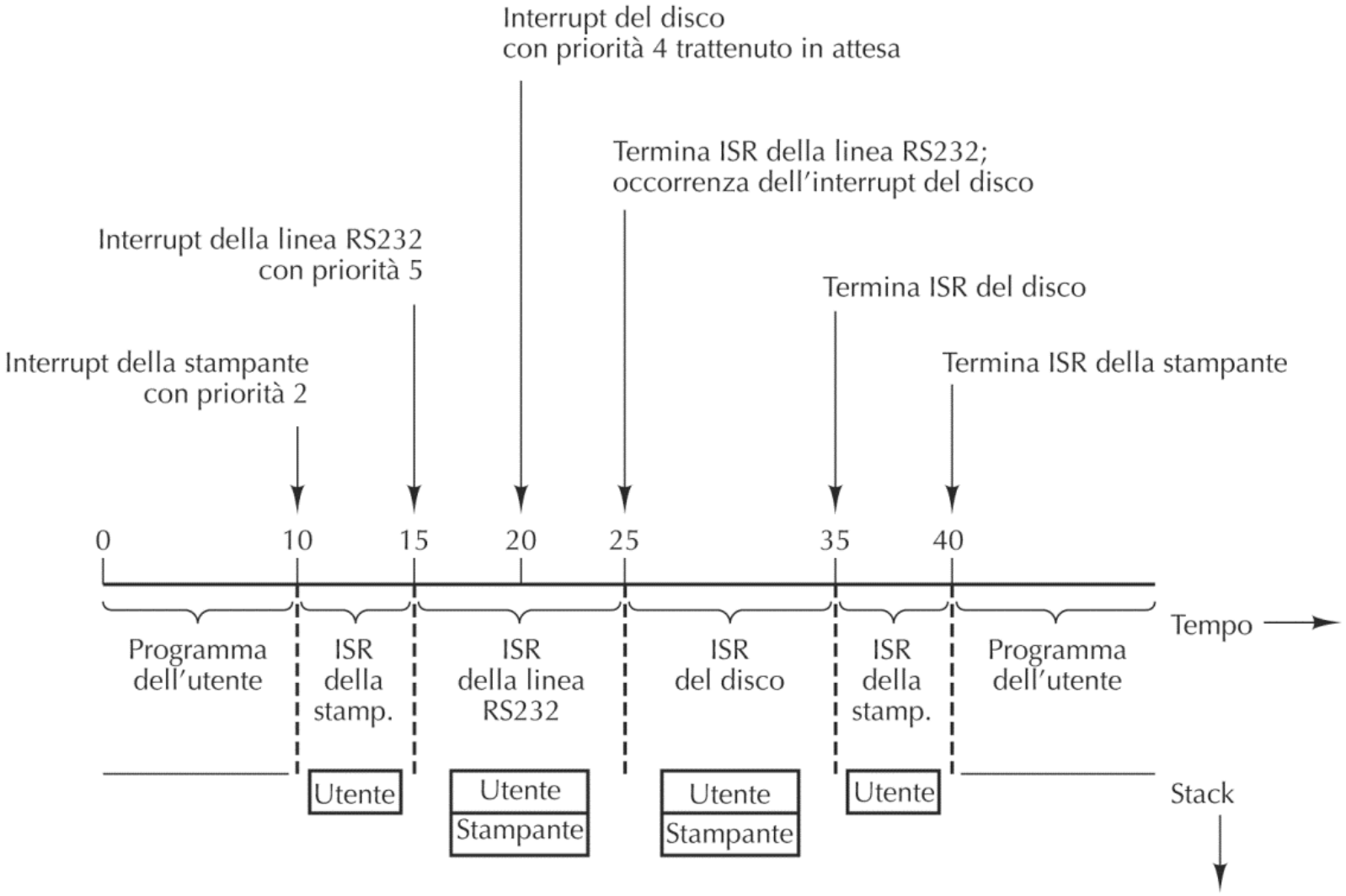

Figura 5.43 Sequenza di interrupt.