

# Lezione E4

## Architettura ARM - II

Sistemi embedded e real-time

18 ottobre 2012

Marco Cesati

Dipartimento di Ingegneria Civile e Ingegneria Informatica  
Università degli Studi di Roma Tor Vergata

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4.1

## Di cosa parliamo in questa lezione?

In questa lezione completiamo la descrizione generale dell'architettura ARM

- 1 Modi di indirizzamento
- 2 Istruzioni aritmetiche
- 3 Istruzioni di copia tra registri
- 4 Istruzioni logiche
- 5 Istruzioni di confronto
- 6 Istruzioni di salto
- 7 Supporto alle procedure
- 8 Gestione delle eccezioni

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4.2

## Modi di indirizzamento con scorrimento

Nei modi di indirizzamento di base e con writeback lo spiazzamento può essere indicato con una costante immediata tra  $-4095$  e  $+4095$

Se invece lo spiazzamento è in un registro è possibile forzare uno scorrimento del suo valore verso sinistra o destra:

- Scorrimento verso sinistra:

```
ldr r1, [r2, r3, lsl #2]   r1 ← [[r2] + 4 × [r3]]
```

```
ldr r1, [r2], -r3, lsl r4  r1 ← [[r2]]
```

```
                           r2 ← r2 - (r3 << (r4 & 31))
```

- Scorrimento logico verso destra:

```
ldr r1, [r2, r3, lsr #1]   r1 ← [[r2] + [r3] >> 1]
```

`lsr #32` è ammesso, mentre `lsl #32` non è ammesso

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4.3

## Caricamento di un valore da 32 bit

*Perché in una architettura RISC è sempre problematico caricare in un registro un valore immediato?*

Perché il valore da caricare ha la stessa dimensione dell'istruzione macchina

Nel linguaggio assembler ARM è possibile utilizzare una espressione simile a questa:

```
ldr r1, =0x12345678
```

In realtà questa viene tradotta dall'assemblatore così:

```
ldr r1, Lcost
```

```
⋮
```

```
Lcost: .word 0x12345678
```

*Non si ha lo stesso problema per caricare l'indirizzo di `Lcost`?*

No: si usa l'indirizzamento relativo al `pc`! `ldr r1, [pc, #offset]`

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4.4

## Istruzioni aritmetiche

La forma base delle istruzioni aritmetiche è

**OP Rd, Rn, Rm**

ove **OP** è il codice (nome) dell'operazione, **Rd** è il registro destinazione, **Rn** e **Rm** sono i registri operandi

Ad esempio:

- L'istruzione **add r2, r1, r0** somma il contenuto dei registri **r1** e **r0** e memorizza il risultato in **r2**
- L'istruzione **sub r10, r4, #520** sottrae al contenuto di **r4** il valore immediato 520 e memorizza il risultato in **r10**

È possibile utilizzare come ultimo operando valori immediati senza segno:

- La costante **#0** non può essere omessa: **add r0, r1** equivale a **add r0, r0, r1**
- L'assemblatore traduce automaticamente una istruzione come **add r0, r1, #-15** in **sub r0, r1, #15**

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4.5

## Istruzioni di somma e sottrazione

- **add** somma  
 $r1 \leftarrow [r2] + [r3]$
- **adc** somma con riporto  
 $r1 \leftarrow [r2] + [r3] + [C]$
- **sub** sottrazione  
 $r1 \leftarrow [r2] - [r3]$
- **sbc** sottrazione con riporto  
 $r1 \leftarrow [r2] - [r3] - [-C]$
- **rsb** sottrazione inversa (valore opposto di **sub**)  
 $r1 \leftarrow [r3] - [r2]$
- **rsc** sottrazione inversa con riporto  
 $r1 \leftarrow [r3] - [r2] - [-C]$

Nella sottrazione il flag **C** è 0 se c'è stato riporto, 1 altrimenti

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4.6

## Valori immediati

Un valore immediato in una operazione aritmetica, logica o di copia è costruito nel seguente modo:

- L'istruzione codifica un valore a 8 bit tra 0 e 255
- Tale valore viene esteso senza segno a 32 bit
- Il valore a 32 bit viene ruotato verso destra di un numero pari di posizioni specificato nell'istruzione (tra 0 e 30)

Esempi:

```
add r1, r2, #1      r1 ← [r2] + 1
add r1, r2, #1, 30  r1 ← [r2] + 4
add r1, r2, #1, 2   r1 ← [r2] + 0x40000000
```

L'assemblatore calcola automaticamente la codifica per una costante immediata a 32 bit

- La codifica migliore è quella con la rotazione più piccola
- Se non esiste codifica si ha un errore di sintassi

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4.7

## Rotazione e scorrimento

Se l'ultimo operando di una istruzione aritmetica è in un registro, allora è possibile:

- scorrerlo logicamente a destra (**lsr**) o sinistra (**lsl**)
- scorrerlo aritmeticamente a destra (**asr**) o sinistra (**asl**)
- ruotarlo verso destra (**ror**)
- ruotarlo verso destra con carry (**rrx**)

Esempi:

```
add r1, r2, r3, lsl #4  scorrimento a sx di 4 posiz.
add r1, r2, r3, ror r4  rotazione a dx di (r4&31) posiz.
add r1, r2, r3, rrx     rotazione a dx di 1 posiz., entra C
```

Non è possibile utilizzare rotazioni e scorrimenti nelle istruzioni di moltiplicazione

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4.8

## I bit di condizione

- Il formato delle istruzioni aritmetiche e logiche consente di specificare in un bit del codice operativo se si devono aggiornare i quattro codici di condizione **Z**, **N**, **C** e **V**

- Esempi:

```
sub r0, r1    non aggiornare i codici di condizione
subs r0, r1   aggiorna i codici di condizione
```

- Il bit di **Carry (C)** è particolarmente utile insieme a **adc** e **sbc** per operare su quantità maggiori di 32 bit:

```
adds r0, r6, r8    (sommano a 64 bit r7:r6 e r9:r8,
adc r1, r7, r9     e memorizzano il risultato in r1:r0)
```

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4.9

## Istruzioni per la moltiplicazione

Esistono più di 20 istruzioni per la moltiplicazione tra interi:

- mul Rd, Rm, Rs**
  - moltiplica i contenuti di **Rm** e **Rs**
  - memorizza i 32 bit inferiori del risultato in **Rd**
- mla Rd, Rm, Rs, Rn**
  - moltiplica i contenuti di **Rm** e **Rs**, e somma il contenuto di **Rn**
  - memorizza i 32 bit inferiori del risultato in **Rd**
- umull/smull Rdlo, Rdhi, Rm, Rs**
  - considera valori senza segno / con segno
  - moltiplica i contenuti di **Rm** e **Rs**
  - memorizza i 64 bit del risultato in **Rdlo** e **Rdhi**
- umlal/smlal Rdlo, Rdhi, Rm, Rs**
- umaal/smaal Rdlo, Rdhi, Rm, Rs**
- Altre istruzioni moltiplicano quantità di 8 e 16 bit, calcolano i 32 bit più significativi di un prodotto con segno, ecc.

*Perché mul e mla non considerano il segno dei numeri?*

I 32 bit inferiori del prodotto sono identici nei due casi  
Ad es.,  $(-1) \times (-3) = 0xffffffffc0000003$

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4.10

## Istruzioni di copia in registro

- mov Rd, Rm** copia il contenuto di **Rm** in **Rd**
- mov Rd, #val** copia il valore immediato in **Rd**
- mvn Rd, Rm** copia l'inverso bit a bit di **Rm** in **Rd**
- mvn Rd, #val** copia l'inverso bit a bit di **val** in **Rd**

È possibile applicare scorrimenti o rotazioni come per le istruzioni aritmetiche, ad esempio:

```
mov r2, r3, lsl #4
```

L'assemblatore traduce una pseudo-istruzione analoga a

```
mov r1, #-17
```

come

```
mvn r1, #16
```

Anche le istruzioni di copia possono impostare i codici condizione (ad es, **movs**)

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4.11

## Istruzioni logiche

- and Rd, Rm, Rn** AND logico bit a bit
  - and r0, r1, r2**  $r0 \leftarrow [r1] \& [r2]$
- eor Rd, Rm, Rn** XOR logico bit a bit
  - eor r0, r1, r2, lsl #2**  $r0 \leftarrow [r1] \wedge ([r2] \ll 2)$
- orr Rd, Rm, Rn** OR logico bit a bit
  - orr r0, r1, #520**  $r0 \leftarrow [r1] \mid 0x208$
- bic Rd, Rm, Rn** Bit clear
  - bic r0, r1, #0xff**  $r0 \leftarrow [r1] \& 0xffffffff00$

Anche le istruzioni logiche possono impostare i codici condizione (ad es, **eors**)

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4.12

## Istruzioni di confronto

- **tst Rn, Rs** Test
  - Effettua l'AND logico bit a bit, imposta i codici di condizione e scarta il risultato
- **teq Rn, Rs** Test Equivalence
  - Effettua l'XOR logico bit a bit, imposta i codici di condizione e scarta il risultato
- **cmp Rn, Rs** Compare
  - Effettua la sottrazione  $[Rn] - [Rs]$ , imposta i codici di condizione e scarta il risultato
- **cmn Rn, Rs** Compare Negated
  - Effettua la somma  $[Rn] + [Rs]$ , imposta i codici di condizione e scarta il risultato

Per l'operando **Rs** è possibile utilizzare valori immediati, scorrimenti o rotazioni

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4.13

## Istruzioni di salto

- Forma generale di una istruzione di salto (branch):  
**bcc locazione**
- **cc** codifica la condizione che deve essere verificata per effettuare il salto
- L'istruzione di salto incondizionato è **b** oppure **bal**
- **locazione** rappresenta l'istruzione alla quale saltare
  - In linguaggio assembler è una etichetta simbolica
  - Nel codice macchina rappresenta uno spiazamento relativo alla posizione dell'istruzione di salto

*Si può saltare senza utilizzare l'istruzione branch bcc? Sì!*

Ad esempio è sufficiente scrivere un valore in **r15** (ossia **pc**)

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4.14

## Calcolo dell'indirizzo destinazione di un salto

- Ogni istruzione di salto codifica un valore a 24 bit in complemento a due
- Si estende con segno tale valore a 32 bit
- Si moltiplica per quattro
- Il risultato è lo spiazamento rispetto al valore di **pc**

*Perché il valore viene moltiplicato per quattro?*

Le istruzioni di 32 bit sono allineate in memoria

*Cosa fa una istruzione di salto il cui valore a 24 bit è -1?*

Non ha alcun effetto sulla sequenza di istruzione eseguite

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4.15

## Istruzioni di salto condizionato

L'esecuzione delle istruzioni di salto condizionato dipende dal valore di alcuni codici di condizione **cc**

CC	CC = 1	CC = 0
Z	<b>beq</b> "=", "= 0"	<b>bne</b> "≠", "≠ 0"
C	<b>bcs</b> carry	<b>bcc</b> no carry
C	<b>bhs</b> "≥ <sub>u</sub> "	<b>blo</b> "< <sub>u</sub> "
N	<b>bmi</b> "< <sub>s</sub> 0"	<b>bpl</b> "≥ <sub>s</sub> 0"
V	<b>bvs</b> overflow	<b>bvc</b> no overflow
$\bar{C} \vee Z$	<b>bls</b> "≤ <sub>u</sub> "	<b>bhi</b> "> <sub>u</sub> "
$N \oplus V$	<b>blt</b> "< <sub>s</sub> "	<b>bge</b> "≥ <sub>s</sub> "
$Z \vee (N \oplus V)$	<b>ble</b> "≤ <sub>s</sub> "	<b>bgt</b> "> <sub>s</sub> "

- Gli operatori con "u" sono di confronto senza segno
- Gli operatori con "s" sono di confronto con segno

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4.16

## Supporto all'invocazione di procedure

- Nei microprocessori **ARM** l'invocazione di una procedura è realizzato tramite l'istruzione **bl** (branch and link)
- **bl** è analoga ad una istruzione di salto (eventualmente condizionata)
- Prima di effettuare il salto, l'indirizzo dell'istruzione successiva a **bl** è salvato nel registro **lr** (o **r14**)

*Non esiste l'istruzione "return": come si termina una procedura?*

Ad esempio: `mov pc, lr`

*In altre architetture l'istruzione "call" salva l'indirizzo di ritorno sullo stack. Che vantaggio ha il metodo usato da ARM?*

Se la procedura non invoca altre sotto-procedure, si evitano due potenziali accessi alla memoria dinamica

D'altra parte la procedura deve salvare sullo stack o in altro registro il contenuto di **lr** prima di invocare altre procedure

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4.17

## Application Binary Interface

- L'**ABI** (Application Binary Interface) definisce gli aspetti dell'ambiente di esecuzione che non sono determinati dall'architettura hardware
  - Come passare i parametri alle procedure
  - Il formato dei file eseguibili
  - Come si gestiscono le eccezioni
  - Come si collegano le librerie
  - Come si interfacciano i debugger
  - ...
- Nel mondo **ARM** sono state utilizzate diverse **ABI**
- La più recente (2005) e maggiormente adottata è

**EABI**: Embedded-Application Binary Interface

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4.18

## Passaggio dei parametri alle procedure

Ruolo speciale dei registri nell'invocazione delle procedure:

<b>pc/r15</b>	Program counter
<b>lr/r14</b>	Link register
<b>sp/r13</b> p	Stack pointer
<b>ip/r12</b>	Registro di appoggio per il linker
<b>fp/r11</b> p	Variabile locale
<b>sl/r10</b> p	Variabile locale
<b>r9</b> p?	Ruolo definito dalla piattaforma
<b>r8</b> p	Variabile locale
<b>r7</b> p	Variabile locale
<b>r6</b> p	Variabile locale
<b>r5</b> p	Variabile locale
<b>r4</b> p	Variabile locale
<b>r3</b>	Argomento # 4
<b>r2</b>	Argomento # 3
<b>r1</b>	Argomento # 2, Risultato (32 bit superiori)
<b>r0</b>	Argomento # 1, Risultato (32 bit inferiori)

Ogni procedura deve preservare il contenuto dei registri "p"

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4.19

## Uso dello stack per salvare e ripristinare i registri

- Poiché una procedura deve preservare il contenuto di alcuni registri, essa deve
  - salvarne il contenuto sullo stack all'inizio della procedura
  - ripristinare il contenuto dallo stack al termine
- È possibile farlo con una serie di istruzioni "push" e "pop":
  - "push" **lr**: `str lr, [sp, #-4]!`
  - "pop" **lr**: `ldr lr, [sp], #4`
- Nei microprocessori **ARM** la tipologia di stack non è pre-determinata
- **EABI** impone l'uso di stack *full descending*:
  - stack crescente per indirizzi decrescenti
  - **sp** puntante all'ultimo elemento inserito
- Tra i registri da preservare c'è anche **lr**, nel caso la procedura debba invocare altre sotto-procedure

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4.20

## Trasferimento di blocchi di registri

- Per facilitare le operazioni di salvataggio e ripristino dei registri sono definite le istruzioni **stmxx** e **ldmxx**
  - Le due lettere **xx** indicano il tipo di stack, ossia le operazioni di aggiornamento del registro puntatore
  - Nel caso di stack **EABI full descending** le lettere sono **fd**
- Esempi:
  - “Push”: `stmfd sp!, {r4-r8, r10-r11, r14}`
  - “Pop”: `ldmfd sp!, {r4-r8, r10-r11, r15}`

*Che effetto particolare si ottiene utilizzando le due istruzioni sopra in una procedura?*

L'istruzione **ldmfd** termina la procedura scrivendo in **pc** (**r15**) l'indirizzo contenuto in **lr** (**r14**)

## Tipi di eccezione

Nei microprocessori **ARM** sono definiti sette tipi di **eccezioni**:

Priorità	Nome	Indir. vettore	Modo
1	<b>Reset</b>	0x00000000	Supervisor
6	<b>Undefined instruction</b>	0x00000004	Undefined
7	<b>Software interrupt</b>	0x00000008	Supervisor
5	<b>Prefetch abort</b>	0x0000000C	Abort
2	<b>Data abort</b>	0x00000010	Abort
4	<b>IRQ (interrupt)</b>	0x00000018	IRQ
3	<b>FIQ (fast interrupt)</b>	0x0000001C	FIQ

Per ciascun tipo di interruzione è definito un **vettore**:

- il **vettore** è in una locazione prefissata della memoria
- il **vettore** codifica la [locazione della] prima istruzione che dovrà gestire l'eccezione

*Perché è definita la priorità delle eccezioni?*

Se due o più eccezioni si verificano contemporaneamente, queste vengono gestite rispettando la loro priorità

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4.21

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4.23

## Modi di esecuzione

In ogni istante il processore **ARM** è in uno dei seguenti stati:

Modo	Esecuzione di	Registri privati
<b>User</b>	applicazioni normali	
<b>System</b>	processi privilegiati	
<b>FIQ</b>	interruzioni veloci	<b>r8 – r14, spsr</b>
<b>IRQ</b>	interruzioni normali	<b>r13, r14, spsr</b>
<b>Supervisor</b>	codice del kernel	<b>r13, r14, spsr</b>
<b>Abort</b>	gestori di fault	<b>r13, r14, spsr</b>
<b>Undefined</b>	emulatori di coprocessori	<b>r13, r14, spsr</b>

In ogni modo di esecuzione sono comunque visibili:

- i 13 registri di uso generale (da **r0** a **r12**)
- r13**  $\equiv$  **sp** (stack pointer), **r14**  $\equiv$  **lr** (link register)
- il contatore di programma **r15**  $\equiv$  **pc**
- un registro di stato (**cpsr**)

**r8 ... r12** sono comuni a tutti i modi di esecuzione tranne **FIQ**: durante una interruzione veloce si possono utilizzare questi registri senza dover salvarne il contenuto in memoria

## Gestione delle eccezioni

All'occorrenza di una **eccezione** si effettuano le seguenti operazioni:

- Il contenuto di **r15** (**pc**) è copiato nel registro **r14** (**lr**) relativo al modo dell'eccezione
- Il contenuto di **cpsr** è copiato nel registro **spsr** relativo al modo dell'eccezione
- I bit di **cpsr** sono modificati per forzare il processore nel modo corrispondente all'eccezione e, se necessario, per disabilitare **FIQ** e/o **IRQ**
- Il registro **pc** (**r15**) è caricato con l'indirizzo specificato dal vettore appropriato

Poiché il registro **r13** (**sp**) è privato per i vari modi di esecuzione (tranne che per **User** e **System**), il gestore dell'eccezione può facilmente utilizzare un proprio stack

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4.22

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4.24

## Terminazione della eccezione

I dettagli per concludere la gestione di una eccezione variano con il tipo di eccezione

- Il riconoscimento di eventi asincroni (FIQ, IRQ) è effettuato al termine dell'esecuzione di ciascuna istruzione
  - pc** (e dunque **lr**) contiene l'indirizzo dell'istruzione conclusa +8
  - Si conclude la gestione tramite **subs pc, lr, #4**
- Quando l'eccezione è sincrona (attivata da una istruzione all'indirizzo  $\mathcal{I}$ )

Tipo di eccezione	lr =	Tipico ritorno
Undefined instruction	$\mathcal{I} + 4$	<b>movs pc, lr</b>
Software interrupt	$\mathcal{I} + 4$	<b>movs pc, lr</b>
Prefetch Abort	$\mathcal{I} + 4$	<b>subs pc, lr, #4</b>
Data Abort	$\mathcal{I} + 8$	<b>subs pc, lr, #8</b>

Quando il target di una istruzione è **pc**, la "s" finale significa: copia il contenuto di **spsr** (modalità dell'eccezione) in **cpsr**

## Manipolazione dei bit di stato

- I gestori delle eccezioni debbono talvolta manipolare lo stato dei bit nel registro **cpsr**, ad esempio per disabilitare le interruzioni
- L'istruzione **mrs** copia il contenuto di **cpsr** o **spsr** in un registro GPR:  
**mrs r1, cpsr**
- L'istruzione **msr** copia il contenuto di un registro GPR in **cpsr** o **spsr**:  
**msr spsr, r0**
- L'istruzione **cps** modifica direttamente la modalità d'esecuzione del processore

*In modalità User **mrs**, **msr** e **cps** non funzionano: perché?*

Se funzionassero qualunque processo (applicazione) potrebbe entrare in modalità privilegiata e scavalcare i meccanismi di protezione del sistema operativo

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4:25

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4:27

## Forme speciali delle istruzioni **stm** e **ldm**

- stmfd sp, {blocco registri}^**
  - Salva i valori dei registri della modalità **User** (a prescindere dalla modalità di esecuzione corrente)
  - sp** non viene modificato
- ldmfd sp[!], {blocco reg. senza pc}^**
  - Carica i valori nei registri della modalità **User** (a prescindere dalla modalità di esecuzione corrente)
- ldmfd sp[!], {blocco registri, pc}^**
  - Carica i valori nei registri della modalità attiva
  - Inoltre copia il contenuto di **spsr** in **cpsr**
- Utili durante la gestione delle eccezioni

## Istruzioni Thumb

Le istruzioni da 32 bit possono essere un problema per sistemi embedded con ridotte capacità di memoria

Molti microprocessori **ARM** dispongono di una ISA alternativa chiamata **Thumb**:

- Istruzioni codificate in 16 bit
- Solo 8 registri GPR: **r0, r1, ..., r7** (oltre a **pc, lr, sp**)
- Molte istruzioni hanno solo un formato a due operandi in cui un registro è sia sorgente che destinazione
- Le uniche istruzioni condizionali sono i salti
- Registri, indirizzi e operandi sono sempre a 32 bit

È possibile mischiare procedure con ISA **ARM** e **Thumb**: il bit **T** in **cpsr** indica l'ISA attiva

Si attiva la modalità **Thumb** con le istruzioni di salto **bx, blx** oppure avendo il bit **T** impostato in **spsr** e copiandolo in **cpsr**

Nell'arch. **ARMv7**: introdotta la ISA **Thumb2** che mescola istruzioni di lunghezza variabile: 16 bit (**Thumb**) e 32 bit (**ARM**)

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4:26

Architettura ARM - II  
Marco Cesati



Schema della lezione  
Modi di indirizzamento  
Istruzioni aritmetiche  
Istruzioni di copia  
Istruzioni logiche  
Istruzioni di confronto  
Istruzioni di salto  
Procedure  
Eccezioni  
ISA Thumb

SERT13 E4:28