Lezione E3

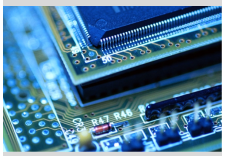
Architettura ARM - I

Sistemi embedded e real-time

11 ottobre 2012

Marco Cesati

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



Schema della lezione
Evoluzione degli ARM
L'architettura ARM
Istruzioni load e store
Modi di indirizzamento

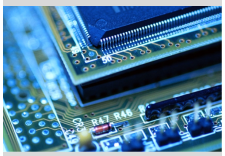
Di cosa parliamo in questa lezione?

In questa lezione iniziamo la descrizione generale dei microprocessori ARM e della relativa architettura

1. Evoluzione degli ARM
2. L'architettura ARM
3. Istruzioni load e store
4. Modi di indirizzamento

Architettura ARM - I

Marco Cesati



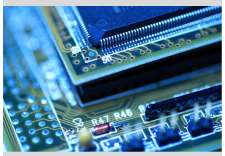
Schema della lezione
Evoluzione degli ARM
L'architettura ARM
Istruzioni load e store
Modi di indirizzamento

Utilizzo di microprocessori nei sistemi embedded

- Spesso i sistemi embedded di fascia alta utilizzano come elementi di calcolo microprocessori sofisticati
- Casi in cui l'uso di un microprocessore è giustificato:
  - sono richieste elevate capacità di calcolo
    - ad es., processamento di segnali audio e video
  - si utilizzano protocolli di comunicazione complessi
    - ad es., rete wireless IEEE 802.11
  - è richiesta memoria di grande capacità
    - ad es., mappe di un navigatore GPS

Cosa hanno in comune gli smartphone basati su iOS (Apple), Symbian (Nokia), Android (Google) e Windows CE (Microsoft)?

Utilizzano tutti lo stesso tipo di microprocessori: ARM

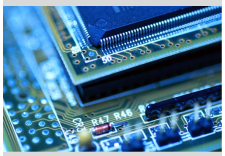


Schema della lezione
Evoluzione degli ARM
L'architettura ARM
Istruzioni load e store
Modi di indirizzamento

Origine dei microprocessori ARM

- Negli anni '80 le società inglesi Acorn e British Broadcasting Corporation (BBC) crearono un nuovo calcolatore personale chiamato BBC Micro
- Grazie a questo accordo, Acorn ebbe le risorse per sviluppare la serie di calcolatori Acorn Archimedes
  - basati su un nuovo microprocessore RISC
  - commercializzati tra il 1987 ed il 1997
  - non ebbero grande successo

Fonte: www.old-computers.com



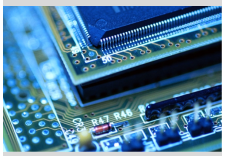
Schema della lezione
Evoluzione degli ARM
L'architettura ARM
Istruzioni load e store
Modi di indirizzamento

Origine dei microprocessori ARM (2)

I calcolatori Archimedes utilizzavano il primo microprocessore RISC commerciale: ARM (Acorn Risc Machine)

- Commercializzato nel 1985
- Al contrario dei calcolatori Archimedes, ebbe subito un immediato successo
- Seguirono l'ARM2 (1985) e l'ARM3 (1989)
- L'Acorn non produceva fisicamente i chip
- La VLSI Technology aveva licenza per fabbricare e commercializzare i chip basati sulle specifiche di Acorn

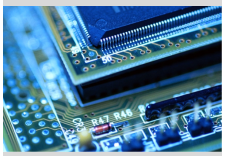
Fonte: www.cpushack.com



Schema della lezione
Evoluzione degli ARM
L'architettura ARM
Istruzioni load e store
Modi di indirizzamento

Evoluzione dei microprocessori ARM

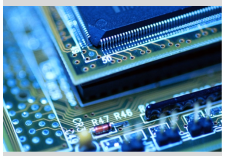
- Il grande successo commerciale degli ARM imponeva di continuare a sviluppare ed estendere l'architettura, ma l'Acorn non aveva abbastanza risorse per farlo
- Nel 1990 venne creata la società Advanced RISC Machines Ltd. a cui partecipavano Acorn, VLSI Technologies e Apple Computer
- Il microprocessore fu ribattezzato Advanced Risc Machine
- Il primo prodotto fu il microprocessore ARM6 (1992), una versione migliorata dell'ARM3
- Da allora l'architettura ARM ha continuato ad evolversi con nuove funzionalità e migliori prestazioni
- Nel mondo embedded gli ARM hanno un successo enorme e sempre crescente:
  - 2005: Nel solo anno sono stati licenziati 1,6 miliardi di chip (di cui 1 miliardo nei telefoni cellulari)
  - 2008: a gennaio raggiunta la soglia di 10 miliardi di chip
  - 2009: si stima che siano il 90% di tutte le CPU RISC a 32 bit
  - 2011: a gennaio raggiunta la soglia di 15 miliardi di chip



Schema della lezione
Evoluzione degli ARM
L'architettura ARM
Istruzioni load e store
Modi di indirizzamento

ARM Intellectual Property

- Uno dei componenti essenziali del successo di ARM consiste nel meccanismo di licenza
- La ARM Ltd. non produce chip ma vende licenze per l'utilizzo dell'IP (Intellectual Property) relativo al progetto di un microprocessore ARM
- La società licenziataria può
  - modificare il progetto per adattarlo ai propri scopi
  - produrre il chip microprocessore
  - integrare il microprocessore all'interno di un proprio sistema (SoC, System on Chip)
  - vendere il progetto modificato ad altre società
- Grazie a questo meccanismo i microprocessori ARM
  - si evolvono rapidamente
  - trovano impiego in molti ambiti e applicazioni

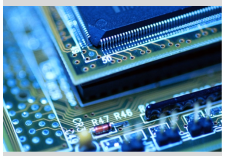


Schema della lezione
Evoluzione degli ARM
L'architettura ARM
Istruzioni load e store
Modi di indirizzamento

Le varianti di microprocessori ARM

Esistono molte varianti di microprocessori ARM, ad esempio:

- Sviluppate da ARM Ltd.:
  - ARM7 (1994)
  - ARM9 (1997)
  - ARM11 (2002)
  - Cortex (2004)
- Sviluppate da altre società licenziatarie:
  - StrongARM (1995): DEC, poi Intel
  - i.MX (2001): Freescale (ex Motorola)
  - XScale (2002): Intel e Marvell
  - OMAP SoC (2007): Texas Instruments
  - Tegra (2008): NVIDIA
  - Snapdragon (2008): Qualcomm
  - Hummingbird (2009): Samsung
  - A4 (2010), A5 (2011) e A5X (2012): Apple
  - Nova e NovaThor (2011): ST-Ericsson



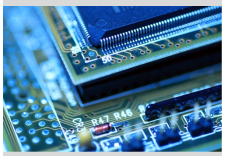
Schema della lezione
Evoluzione degli ARM
L'architettura ARM
Istruzioni load e store
Modi di indirizzamento

Le principali famiglie di microprocessori ARM

Famiglia	Novità	Cache (KB)	MIPS @ MHz
ARM1	pipeline 3 stadi	–	?
ARM2	MMU, GPU, I/O	–	7 @ 12
ARM3	cache	4	12 @ 25
ARM6	indirizzi 32 bit, FPU	4	28 @ 33
ARM7	integrato in SoC	8	60 @ 60
ARM8	pipeline 5 st., pred. salti	8	84 @ 72
ARM9	architettura Harvard	16+16	300 @ 300
ARM9E	istr. DSP migliorate	16+16	220 @ 200
ARM10E	pipeline 6 st.	32+32	500 @ 400
ARM11	pipeline 9 st.	variabile	740 @ 665
Cortex	pipeline supersc. 13 st.	variabile	2000 @ 1000
XScale	pipeline 7 st.	L1: 32+32 L2: 512	1000 @ 1250

Fonte: W. Stallings, Architettura ed organizzazione dei calcolatori, 8 ed., Pearson, 2010

Le caratteristiche possono variare a seconda del modello nella stessa famiglia



Schema della lezione
Evoluzione degli ARM
L'architettura ARM
Istruzioni load e store
Modi di indirizzamento

La famiglia Cortex

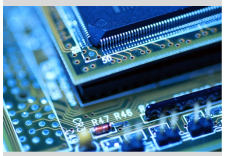
La famiglia Cortex contiene modelli ARM adatti agli usi più diversi

Sono stati definiti tre diversi profili applicativi:

- Cortex-A: profilo "Applications" per sistemi di uso generale
  - ad es., smartphone, TV digitali
- Cortex-R: profilo "Real-time" per sistemi real-time
  - ad es., impianti frenanti, dischi rigidi, switch di rete
- Cortex-M: profilo "Microcontroller" per sistemi embedded
  - ad es., sensori intelligenti, calcolatrici, pacemaker

Le famiglie di microprocessori ARM hanno essenzialmente lo stesso insieme di base di istruzioni macchina

Tuttavia programmi compilati specificatamente per un certo microprocessore non funzionano necessariamente con un microprocessore di un'altra famiglia

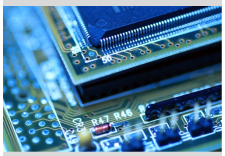


Schema della lezione
Evoluzione degli ARM
L'architettura ARM
Istruzioni load e store
Modi di indirizzamento

Le versioni dell'architettura ARM

ISA	Famiglia
ARMv1	ARM1
ARMv2	ARM2, ARM3
ARMv3	ARM6, ARM7
ARMv4	StrongARM, ARM7TDMI, ARM9TDMI
ARMv5	ARM7EJ, ARM9E, ARM10E, XScale
ARMv6	ARM11, Cortex-M
ARMv7	Cortex-A, Cortex-R
ARMv8	(core a 64 bit, non ancora commercializzati)

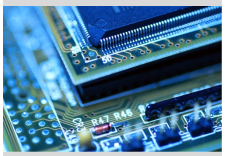
- ARMxTx: insieme alternativo di istruzioni Thumb a 16 bit
- ARMxDx: supporto per debug via JTAG
- ARMxMx: unità moltiplicazione più efficiente
- ARMxIx: supporto per debug con *EmbeddedICE*
- ARMxEx: supporto per DSP e multimedia (implica TDMI)
- ARMxJx: supporto per Java bytecode nativo (Jazelle)



Schema della lezione
Evoluzione degli ARM
L'architettura ARM
Istruzioni load e store
Modi di indirizzamento

Caratteristiche principali dei microprocessori ARM

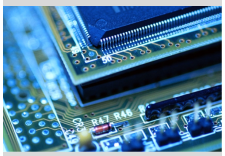
- Architettura a 32 bit
  - Dimensione dei registri e dei dati su cui operano le istruzioni
- Essenzialmente architettura RISC
  - Lunghezza fissa delle istruzioni macchina: 32 bit
  - Operazioni compiute sui registri, non in memoria
  - Istruzioni "load" e "store" per accedere alla memoria
- Memoria indirizzabile al singolo byte, con indirizzi da 32 bit
- Accesso alla memoria "allineato"
  - I dati in memoria sono lunghi 8 bit, 16 bit o 32 bit
  - L'indirizzo di un dato a 16 bit deve essere multiplo di 2
  - L'indirizzo di un dato a 32 bit deve essere multiplo di 4
- Memorizzazione di tipo "little-endian" o "big-endian"
  - A seconda dello stato di una linea di input del chip



Schema della lezione
Evoluzione degli ARM
L'architettura ARM
Istruzioni load e store
Modi di indirizzamento

Caratteristiche notevoli dei microprocessori ARM

- Esecuzione condizionale delle istruzioni
  - Molte istruzioni macchina possono essere eseguite o meno a seconda del valore dei bit di stato
  - In molte altre architetture solo i salti sono condizionali
- Assenza di istruzioni esplicite per lo scorrimento dei bit
  - Tuttavia nelle istruzioni logiche, aritmetiche e copia è possibile eseguire lo scorrimento dei bit di un operando
- Supporto per diverse operazioni di moltiplicazione
  - Varianti ottimizzate per l'elaborazione dei segnali
- Non esiste una istruzione macchina per la divisione
  - Deve essere realizzata da una procedura software ovvero affidata ad un coprocessore
- Schemi di indirizzamento tipici di una architettura CISC
  - Auto-incremento e auto-decremento degli indirizzi
  - Indirizzamento relativo al *program counter*
  - Una singola istruzione può trasferire dati tra un blocco di memoria ed un insieme di registri

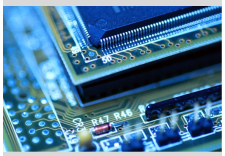


Schema della lezione
Evoluzione degli ARM
L'architettura ARM
Istruzioni load e store
Modi di indirizzamento

Il banco dei registri

- L'ISA di un ARM definisce 31 registri a 32 bit utilizzabili in ogni istruzione macchina
- Ad ogni istante sono visibili solo 16 registri chiamati

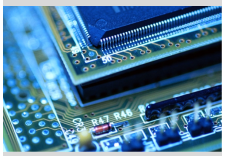
  `r0`, `r1`, `r2`, `r3`, `r4`, `r5`, `r6`, `r7`, . . . , `r13`, `r14`, `r15`
- `r10` (`sl`) talvolta contiene la dimensione dello stack
- `r11` (`fp`) è utilizzato spesso come *frame pointer*
- `r12` (`ip`) può essere utilizzato per l'invocazione di procedure
- `r13` (`sp`) è utilizzato spesso come *stack pointer*
- `r14` (`lr`) è il *link register*: può contenere l'indirizzo di ritorno di una funzione
- `r15` (`pc`) è il *program counter*: memorizza la posizione nel programma in esecuzione
  - `r15` contiene l'indirizzo della 2^a istruzione sotto a quella in esecuzione



Schema della lezione
Evoluzione degli ARM
L'architettura ARM
Istruzioni load e store
Modi di indirizzamento

I registri di stato

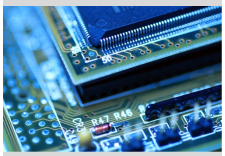
- Il registro di stato principale è chiamato **`cpsr`** (Current Program Status Register)
- In **`cpsr`** sono contenuti quattro bit di condizione:
  - N risultato negativo
  - Z risultato nullo (zero)
  - C si è verificato un riporto (carry)
  - V si è verificato un trabocco (overflow)
- In **`cpsr`** sono contenuti anche alcuni bit di controllo:
  - due bit per disabilitare le interruzioni
  - cinque bit che codificano il modo corrente del processore
  - due bit T e J che specificano la ISA utilizzata (ARM, Thumb, Jazelle)
- Ulteriori bit di condizione e di controllo sono definiti nelle architetture ARMv5 e ARMv6
- Esistono anche cinque registri **`spsr`** (Saved Program Status Register) utilizzati per preservare il valore in **`cpsr`** al verificarsi di una eccezione



Schema della lezione
Evoluzione degli ARM
L'architettura ARM
Istruzioni load e store
Modi di indirizzamento

Tipi di istruzioni

- Istruzioni "load" e "store"
- Istruzioni aritmetiche
- Istruzioni di copia
- Istruzioni logiche
- Istruzioni di confronto
- Istruzioni di salto



Schema della lezione
Evoluzione degli ARM
L'architettura ARM
Istruzioni load e store
Modi di indirizzamento

Istruzioni “load” e “store”

Poiché le architetture ARM sono RISC, due soli tipi di istruzione trasferiscono dati da e verso la memoria:

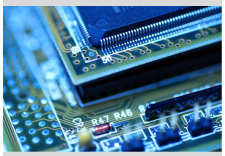
- L’istruzione di “load” (lettura dalla memoria) è **`ldr`**
- L’istruzione di “store” (scrittura in memoria) è **`str`**

È possibile indicare l’indirizzo della cella di memoria come parametro immediato (costante) di queste istruzioni?

Gli indirizzi sono di 32 bit, e le istruzioni hanno un formato fisso di 32 bit: in generale non è possibile!

L’indirizzo deve essere contenuto o derivabile dal contenuto di un registro generale

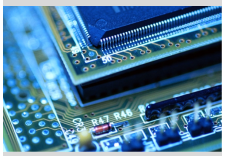
Ad esempio: **`str r2,[r4]`** scrive in memoria il valore a 32 bit contenuto in **`r2`** iniziando dall’indirizzo contenuto in **`r4`**



Schema della lezione
Evoluzione degli ARM
L’architettura ARM
Istruzioni load e store
Modi di indirizzamento

Trasferimento di interi a 8, 16 e 32 bit

- Le istruzioni **`ldr`** e **`str`** trasferiscono interi a 32 bit da/verso locazioni di memoria allineate a multipli di 4
- Le istruzioni **`ldrb`**, **`ldrsb`** e **`strb`** trasferiscono interi a 8 bit da/verso un singolo byte di memoria
  - **`ldrb`** memorizza il valore letto dalla memoria in un registro a 32 bit estendendolo con zeri
  - **`ldrsb`** memorizza il valore letto dalla memoria in un registro a 32 bit estendendolo con il segno
- Le istruzioni **`ldrh`**, **`ldrsh`** e **`strh`** trasferiscono interi a 16 bit da/verso locazioni di memoria allineate a multipli di 2
  - **`ldrh`** memorizza il valore letto dalla memoria in un registro a 32 bit estendendolo con zeri
  - **`ldrsh`** memorizza il valore letto dalla memoria in un registro a 32 bit estendendolo con il segno



Schema della lezione
Evoluzione degli ARM
L’architettura ARM
Istruzioni load e store
Modi di indirizzamento

Modi di indirizzamento di base

- Pre-indexed mode: l'indirizzo effettivo è ottenuto sommando o sottraendo il contenuto di un registro base e di uno spiazzamento immediato o in registro

```
ldr r1,[r2,#-100]      r1 ← [[r2] − 100]
ldr r3,[r5,r7]         r3 ← [[r5] + [r7]]
ldr r2,[r2,-r1]        r2 ← [[r2] − [r1]]
ldr r12,[r11]          r12 ← [[r11]]
```

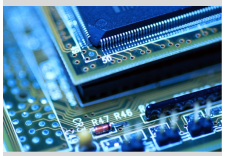
- Relative addressing mode: come il precedente, ma il registro base è `r15` (ossia il *program counter* `pc`)

  Ad esempio, le istruzioni assembler

```
DATO:   .word 100
        ldr r1, DATO
```

  sono tradotte in

```
        .word 0x00000064
        ldr r1,[pc,#-12]
```

Schema della lezione
Evoluzione degli ARM
L'architettura ARM
Istruzioni load e store
Modi di indirizzamento

Modi di indirizzamento con writeback

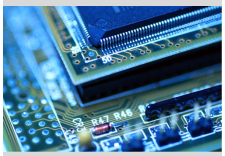
- Pre-indexed with writeback mode: come il modo pre-indexed, tranne che il registro base viene aggiornato con l'indirizzo effettivo

```
ldr r1,[r2,#-100]!     r1 ← [[r2] − 100]
                       r2 ← [r2] − 100
```

- Post-indexed: l'indirizzo effettivo è quello nel registro base; poi l'indirizzo base è aggiornato sommando o sottraendo uno spiazzamento immediato o in registro

```
ldr r1,[r2],#-100      r1 ← [[r2]]
                       r2 ← [r2] − 100

ldr r1,[r2],r3         r1 ← [[r2]]
                       r2 ← [r2] + [r3]
```

Schema della lezione
Evoluzione degli ARM
L'architettura ARM
Istruzioni load e store
Modi di indirizzamento