Architetture degli Elaboratori

1 Rappresentazione dei numeri

Sistemi di numerazione

- **Binario**: solo 0 e 1. Usato per rappresentare dati nel computer.
- **Ottale (base 8)**: cifre da 0 a 7. Ogni cifra corrisponde a 3 bit.
- **Esadecimale (base 16)**: cifre da 0–9 e lettere A–F. Ogni cifra rappresenta 4 bit.

Conversione binario-decimale

Somma delle potenze di 2. Esempio: $1010_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 10_{10}$.

Complemento a 1

- Inverti tutti i bit e poi converto in decimale (uno all'inizio è negativo)

Complemento a 2

- Convertiamo in binario ma la prima cifra la moltiplichiamo per (-2^{n-1})

Overflow

Accade quando il risultato di un'operazione supera l'intervallo rappresentabile. Es: $1111_2 + 0001_2 = 0000_2$ (con overflow).

Floating Point - IEEE 754

- Rappresentazione in forma $\pm mantissa \times 2^{esponente}$.
- **Single precision**: 32 bit = 1 (segno) + 8 (esponente) + 23 (mantissa).
- Problemi: non associatività, precisione finita, presenza di NaN, $\pm\infty$.

2 Algebra di Boole e porte logiche

Porte fondamentali

NOT:

In	Out
0	1
1	0

AND:

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

OR:

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

XOR:

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Porte derivate

- **NAND**: NOT(AND), universale.
- **NOR**: NOT(OR), universale.
- **XNOR**: NOT(XOR).

Leggi algebriche

- $\neg(x \wedge y) = \neg x \vee \neg y$ (De Morgan)
- Idempotenza, assorbimento, distributività

3 Circuiti aritmetici combinatori

Half-Adder

Somma due bit:

- Somma = A XOR B
- Carry = A AND B

Full-Adder

Somma A, B e Carry-in:

- Somma = A XOR B XOR C_{in}
- Carry = $AB + AC_{in} + BC_{in}$

Ripple-Carry Adder

Serie di full-adder per sommare numeri a più bit. Ogni carry-out è il carry-in successivo.

4 Logica sequenziale: latch e flip-flop

SR Latch

Due ingressi: S (set) e R (reset). Stato instabile se entrambi a 1.

Gated D Latch

Memorizza valore D quando Enable è attivo.

D Flip-Flop

Salva D solo sul fronte di salita del clock. Base della memoria sequenziale nei processori.

5 Multiplexer, Decoder e Demultiplexer

Multiplexer

Seleziona uno tra N input. Richiede $\log_2 N$ bit di selezione.

Decoder

Trasforma input binario in uno tra 2^n segnali di uscita.

Demultiplexer

Opposto del multiplexer: 1 input distribuito a N uscite selezionate.

6 ALU, Register File, Control Unit e PC

Register File

Insieme di registri indirizzati. Usati da ALU e istruzioni.

ALU

Esegue operazioni logico-aritmetiche (ADD, AND, OR, SUB...).

Control Unit

Decodifica l'istruzione e imposta i segnali.

Program Counter (PC)

Tiene traccia dell'indirizzo dell'istruzione da eseguire.

7 Assembly ARM (ARM-v7a)

Struttura generale

ARM è una famiglia di processori RISC. L'assembly ARM-v7a lavora con 13 registri generali (R0–R12), uno stack pointer (R13), un link register (R14) e un program counter (R15).

- **R0–R3**: passaggio parametri e valori generici.
- **R13 (SP)**: stack pointer.
- **R14 (LR)**: contiene l'indirizzo di ritorno nelle chiamate di funzione.
- **R15 (PC)**: program counter, indica l'indirizzo dell'istruzione corrente.
- **CPSR (PSR)**: contiene i flag (Negative, Zero, Carry, Overflow) e controlla lo stato del processore.

Istruzioni fondamentali

- `MOV Rn, #valore`: carica un valore costante in un registro.
- `ADD Rd, Rn, Rm`: somma i registri Rn e Rm, salva in Rd.
- `SUB Rd, Rn, Rm`: sottrae Rm da Rn.
- `CMP Rn, Rm`: confronta i due registri, aggiornando i flag nel PSR.
- `B etichetta`: salto incondizionato.
- `BEQ/BNE/BGE/BLT`: salti condizionati (eseguiti in base al risultato della CMP).
- `LDR Rd, [Rn]`: carica in Rd il contenuto dell'indirizzo puntato da Rn.
- `STR Rm, [Rn]`: salva in memoria all'indirizzo di Rn il contenuto di Rm.

Esempio: sommare i numeri da 1 a 10

```
MOV R0, #1          ; contatore i = 1
MOV R1, #0          ; accumulatore somma = 0
loop:
  ADD R1, R1, R0    ; somma += i
  ADD R0, R0, #1    ; i++
  CMP R0, #11
  BNE loop          ; se i != 11, continua il ciclo
```

8 Stack e chiamate a funzione

Struttura dello stack

Lo stack cresce verso indirizzi più bassi. Il registro R13 (SP) punta all'ultima posizione utilizzata.

Chiamate a funzione in ARM

- `BL funzione`: chiama la funzione salvando il ritorno in R14 (LR).
- `BX LR`: ritorna dalla funzione usando il valore di LR.
- `PUSH {R4, LR}`: salva sullo stack R4 e LR.
- `POP {R4, LR}`: ripristina i registri.

Esempio: chiamata a funzione con salvataggio dello stato

```
main:
  MOV R0, #5
  PUSH {R4, LR}
  BL funzione
  POP {R4, LR}
  BX LR
funzione:
  ADD R0, R0, #1
  BX LR
```

9 I/O e Interrupt

Port Mapped I/O (PMIO)

Tecnica in cui i dispositivi di I/O hanno un indirizzamento separato rispetto alla memoria. Si usano istruzioni speciali (IN/OUT) per accedere ai dispositivi.

Memory Mapped I/O (MMIO)

Dispositivi mappati all'interno dello spazio di indirizzamento della memoria. Le periferiche si gestiscono con le stesse istruzioni di accesso alla memoria (LDR/STR).

Polling vs Interrupt

- **Polling**: la CPU controlla periodicamente se un dispositivo è pronto.
- **Interrupt**: il dispositivo segnala alla CPU quando ha bisogno di attenzione, interrompendo il flusso normale di esecuzione.

DMA (Direct Memory Access)

Permette a un dispositivo di trasferire dati direttamente da/verso la memoria senza coinvolgere la CPU. La CPU configura il controller DMA e viene interrotta solo a operazione completata.

10 Memoria virtuale

Spazio degli indirizzi

Ogni processo vede uno spazio virtuale indipendente. Gli indirizzi virtuali devono essere tradotti in indirizzi fisici.

MMU (Memory Management Unit)

Hardware che effettua la traduzione da indirizzi virtuali a indirizzi fisici tramite una **page table**.

Paging

- Divide lo spazio di indirizzi virtuali in blocchi fissi (es. 4KB).
- Ogni pagina virtuale viene mappata a un frame fisico.

TLB (Translation Lookaside Buffer)

Cache dei mapping recenti per velocizzare la traduzione.