**Parte I - Capitolo 2 - Le Istruzioni: Il Linguaggio Dei Calcolatori**

Per impartire comandi all'hardware di un calcolatore bisogna comunicare nel suo linguaggio attraverso un *instruction set* cioè un dizionario di comandi comprendibili dall'architettura del calcolatore. L'insieme di istruzioni prescelto proviene dal processore MIPS, ma saranno esaminati anche i processori ARM, molto simili al MIPS, e il processore Intel x86.

Qualsiasi calcolatore deve sapere eseguire le operazioni aritmetiche. La notazione del linguaggio assembler MIPS per calcolare la somma è: *addi a,b,c* (a=b+c). I commenti devono essere preceduti dal simbolo #. Tutte le istruzioni in linguaggio assembler hanno esattamente tre operandi in modo da mantenere l'hardware semplice come illustrato nel ***Principio Di Progettazione Numero 1: la semplicità favorisce la regolarità***. Per il salvataggio di variabili si usano i registri. La dimensione dei registri è di 32 bit; un gruppo di 32 bit è molto frequente in MIPS e viene indicato con il termine *word*. Anche i registri, tipicamente, sono 32, da cui si può enunciare il ***Principio Di Progettazione Numero 2: minori sono le dimensioni, maggiore è la velocità***. Questo principio non è valido in valore assoluto in quanto 31 registri non sono necessariamente più veloci di 32, ma si utilizzano 32 registri perchè 32 è il numero di bit necessario per indirizzarli all'interno dell'istruzione. Vengono utilizzati i registri $s0,$s1... per le veriabili e i registri $t0,$t1... come registri temporanei.

Anche se il processore può contenere un numero limitato di dati, la memoria può contenere miliardi di dati. Nel linguaggio assembler MIPS le istruzioni richiedono che gli operandi siano memorizzati nei registri, quindi l'assembler deve contenere le istruzioni di trasferimento dati dalla memoria ai registri. L'istruzione di trasferimento *load* sposta un dato dalla memoria al registro; l'istruzione complementare è lo *store* che trasferisce i dati dal registro alla memoria.

Nel linguaggio MIPS c'è spesso bisogno di costanti o operandi immediati (addi,subi): in questo modo il programma risulta più veloce e impiega meno energia elettrica, da qui il ***Principio Di Progettazione Numero 3: rendi più veloce possibile il caso più comune***. I numeri vengono rappresentati in base 2, quindi le informazioni sono contenute da stringhe di bit. E' facile intuire che è possibile 2^32 diverse combinazioni di numeri su cui si possono eseguire operazioni eseguite in modulo e segno o complemento a due. Anche le istruzioni sono memorizzate nel calcolatore come sequenze di segnali elettrici rappresentati da bit. Per eseguire il codice, vengono inviati stringhe di 32 bit alla macchina che le esegue. I 32 bit vengono divisi in campi con significato diverso: *op* (operazione base dell'istruzione, 6bit), *rs* (registro contentente il primo operando sorgente, 5bit), *rt* (registro contente il secondo operando sorgente, 5bit), *rd* (registro destinazione, riceve il risultato, 5bit), *shamt* (numero di posizioni di shift, 5bit), *funct* (funzione, 6bit). Questa rappresentazione è un compromesso scelto dai progettisti MIPS, cioè quello di mantenere uguale la lunghezza di tutte le istruzioni. Il formato descritto è di tipo R o formato R (registro) e aiuta ad esplicitare il ***Principio Di Progettazione Numero 4: un buon progetto richiede buoni compromessi***. Un formato differente è quello di tipo I (immediato) in cui i bit sono divisi in: *op* (6bit), *rs* (5bit), *rt* (5bit), *costante o indirizzo* (16bit).

Oltre alle operazioni aritmetiche, il calcolatore ha bisogno di operare anche su gruppi di bit o parole. Vengono quindi definite le operazioni logiche. Un'operazione di *shift* permette di spostare tutti i bit di una parola a destra o a sinistra, riempiendo gli spazi vuoti con degli zeri. Un'operazione di *AND* è un'operazione bit a bit che scrive un risultato 1 solo se entrambi gli operandi hanno un 1 in quella stessa posizione. Questa operazione viene tradizionalmente usata per le maschere. Un'operazione di *OR* è un'operazione bit a bit che inserisce 1 nel risultato nella posizione in cui almeno un operando è uguale a 1. Un'operazione di *NOT* inverte il bit dell'operando nel risultato.

I linguaggi di programmazione ad alto livello presentano processi decisionali attraverso il costrutto if/else. Per quanto riguarda la programmazione in linguaggio assembler, le decisioni vengono prese attraverso le istruzioni di *salti condizionati* (*branch*), che riflettono il comando if ad alto livello, e le istruzioni di *salti incondizionati* (*jump*), che riflettono il comando else della programmazione ad alto livello. Prendere decisioni è importante sia per quanto riguarda il costrutto if/else, sia per iterare un certo calcolo più volte come avviene nei cicli. I test per verificare l'uguaglianza o la disuguaglianza sono i più diffusi e sono alla base dei cicli for. Per questo risultano utili i comandi in linguaggio assembler di *set* che restituisce un risultato booleano (0 o 1) in una variabile dopo un determinato confronto (equal, not equal, less). Per il costrutto switch/case è possibile utilizzare una sequenza di condizioni if/else.

Le procedure, o funzioni, costituiscono uno degli strumenti più utilizzati dai programmatori per organizzare i programmi. MIPS fa uso di 32 registri suddivisi in questo modo: $a0-$a3 per il passaggio dei parametri, $v0-$v1 per la restituzione di valori calcolati, $t0-$t9 per i registri temporanei, $s0-$s7 per i registri salvati, $gp global pointer per i dati statici, $sp stack pointer puntatore allo stack, $fp frame pointer, $ra return address. Il MIPS supporta un salto incondizionato jr per il salto a un registro specificato (jr $ra).

- Procedure Foglia

- Allocazione sullo stack e sullo heap

- Comunicare con le persone (stringhe)

- Compilatore

- Linker

- Loader

- Architettura ARM

- Architettura x86