Indice

Appunti ACSO

I principi fondamentali

I processori eseguono le istruzioni in maniera sequenziale (non vedremo la Out of Order Execution). Il processore contiene una sezione di controllo che controlla l'esecuzione leggendo le istruzioni dalla memoria. I vari programmi da eseguire sono salvati in una memoria.

L'architettura di riferimento è l'architettura di Von Neumann. Esso contiene:

- Processore
  - Sezione di controllo
  - ALU
  - Registri
- Memoria
- Interfacce I/O
- Bus che collega i vari componenti sopra elencati

I registri sono delle piccole celle di memoria che il processore usa per salvare dati utili. Ogni registro ha una dimensione massima specificata in bit. I registri che studieremo saranno quelli a 32 bit.

Viene detta parola del processore la lunghezza dei suoi registri, ossia la lunghezza massima della sequenza di bit che può gestire.

Esistono dei registri particolare:

- PC (Program Counter): contiene l'indirizzo della prossima istruzione in memoria
- IR (Instruction Register): contiene l'intera istruzione da eseguire

Il ciclo di esecuzione di un'istruzione è divisa nelle seguenti fasi:

- fetch: viene presa dalla memoria attraverso il bus l'istruzione puntata dal program counter e incrementalo nel frattempo (memoria più lenta del processore)
- decode: decodifica l'istruzione e prepara l'esecuzione prelevando gli operandi
- execute: esegui e memorizza i risultati

I vari processori si differenziano in base alla lunghezza delle istruzioni che usano (può essere una singola parola (RISC) oppure più (CISC)).

Data path

Il datapath è formato da registri e da ALU. I registri sono divisi in:

- utilizzabili dall'assembler
- di supporto per altri componenti (es.: 3 registri di support dell'ALU (sorgente, sorgente, destinazione))

Abbiamo due classi di istruzioni:

- Prendono registri e memorizzano in altri registri (registro-registro)
- Gestiscono lo spostamento da registri a memoria (registro-memoria)

L'esecuzione di una istruzione registro-registro viene detta ciclo di data path. Il ciclo di data path è collegato al ciclo di clock.

Codifica delle istruzioni

Ogni istruzione è divisa in codice operativo e campi (field). Il codice operativo indica il tipo di operazione. I campi vengono usati per gli operandi dell'istruzione.

Le modalità di indirizzamento descrivono le diverse modalità attraverso le cui far riferimento agli operandi nelle istruzioni.

Architettura LOAD/STORE

Il numero di registri ad uso generale non è abbastanza grande per mantenere tutte le variabili di un programma. Ad ogni variabile viene quindi assegnata una locazione in memoria nella quale salvare il contenuto del registro che la rappresenta quando deve essere usato per altro.

Poiché gli operandi delle istruzioni possono provenire solamente dai registri ad uso generale, servono delle istruzioni di caricamento e salvataggio a memoria. Da qui il nome dell'architettura.

Interruzioni (interrupt)

Il normale flusso dei programmi può essere interrotto attraverso le interruzioni. Quando un dispositivo esterno vuole richiedere l'attenzione del processore, esso attiva un segnale hardware (segnale di interrupt) per notificarlo.

Il check degli interrupt viene fatto alla fine dell'esecuzione di un'istruzione ma prima del fetch della successiva. Se viene trovato un segnale di interrupt, il processore interrompe la normale esecuzione del programma ed esegue la richiesta. Una volta gestita il processore torna ad eseguire normalmente il suo programma.

La gerarchia di memoria

E' divisa in una gerarchia (dalla più veloce alla più lenta):

- Registri
- Cache (L1, L2, L3)
- Memoria fisica: RAM (es. DDR-SDRAM)
- Memoria a stato solid (SSD)
- Memoria virtuale: basata su file (HDD)

Indirizzi di memoria principale

La memoria principale è suddivisa in celle. La dimensione di una cella è chiamata parola, come anche il numero massimo di bit che il processore può gestire. La parola di memoria non per forza avrà la stessa dimensione della parola del processore.

Nel MIPS useremo una parola di memoria di 8 bit (memoria indirizzabile al byte). Di conseguenza una parola MIPS è lunga 4 parole di memoria.

Il tempo di indirizzamento di ogni singola parola è uguale.

L'insieme di tutte le celle di memoria indirizzabili con 1 parola di processore è detto spazio di memoria: per una parola di 32 bit si ha uno spazio di memoria di 4 GB.

Bus di sistema

Un bus è un insieme di fili. Ogni filo trasferisce un bit. I vari fili sono suddivisi in categorie:

- Bus dati: comprende le linee usate per trasferire dati da/verso memoria. La dimensione del bus dati deve essere abbastanza da garantire il trasferimento contemporaneo di una o più parole di memoria
- Bus indirizzi: comprende le linee sulle quali la CPU procede a trasmettere gli indirizzi di memoria delle risorse
- Bus di controllo: comprende le linee su cui transitano le informazioni ausiliarie per la corretta definizione delle operazioni e per sincronizzare CPU e memoria

Il bus può essere utilizzato per un solo trasferimento alla volta: in ogni istante solo due entità possono comunicare (master - slave). Il master è il processore, gli slave sono le periferiche. Il processore regola l'accesso delle varie periferiche. L'unico filo che può essere usato senza permesso del processore è quello di interrupt request.

L'architettura a bus singolo è molto flessibile e semplice. I dispositivi, però, hanno velocità diverse. Serve, quindi, un meccanismo di sincronizzazione tra le varie periferiche.

Esempio: operazione di lettura da memoria

1. CPU fornisce l'indirizzo della parola desiderata sul bus indirizzi e richiede la lettura sul bus di controllo
2. Quando la memoria ha completato la lettura della parola richiesta, il dato viene trasmesso sul bus dati

MIPS assembler

Il linguaggio assembler è simbolico. E' il primo livello di astrazione prima del linguaggio macchina.

Un programma è composto da più file (oggetti). Il compito di "collegare" i vari oggetti è delegato al linker.

L'assembler non istruzioni che effettuano un controllo di flusso come inteso in C. L'unico strumento che ha è il jump/branch che permette di saltare da un'istruzione all'altra.

Sintassi

Nel MIPS i registri sono numerati da 0 a 31 e hanno dei nomi simbolici. Vengono identificati dal simbolo `$`. Mettere un registro tra parentesi tonde effettua un'operazione di indirezione. Il primo registro nei field è quasi sempre la destinazione del risultato dell'istruzione.

Direttive I comandi che iniziano con `.` sono direttive del preprocessore. Esse sono:

- `.text` o `.data`: indica che le linee successive sono istruzioni o dati
- `.align n`: specifica l'allineamento a 2^n bit
- `.globl main`: indica che l'etichetta `main` è visibile anche in altri file (visibilità globale)
- `.aciiz`: specifica un'area di memoria che contiene una stringa ASCII terminata da `\0`
- `.space n`: riserva uno spazio di n byte

Registri Essi sono 32, numerati da 0 a 31 (occupa 5 bit). Ogni istruzione, quindi, può usare 3 registri alla volta. Gli operandi delle istruzioni devono essere prima salvati nei registri.

Registri referenziabili:

- `$0`: contiene sempre 0
- `$1`/`$at`: riservato
- `$v0` e `$v1`: usati per risultati di funzioni e calcolo espressioni
- `$a0` - `$a3`: usati per il passaggio di argomenti
- `$t0` - `$t7`: variabili temporanee
- `$s0` - `$s7`: variabili da preservare
- `$k0` e `$k1`: riservati al kernel
- `$gp`: global pointer (punta ad area di dati globale/statica)
- `$sp`: stack pointer
- `$fp`: frame pointer (puntatore ai frame funzione; in aiuto a `$sp`)
- `$ra`: return address utilizzato nelle chiamate di funzione

Registri non referenziabili:

- `$pc`: program counter
- `$hi`: registro per moltiplicazione/divisione
- `$lo`: registro per moltiplicazione/divisione

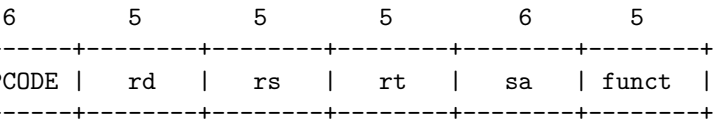
Istruzioni Le istruzioni si dividono in 4 categorie:

1. istruzioni aritmetico logiche
2. istruzioni di trasferimento da/verso memoria (ad es. `lw` e `sw`)
3. istruzioni di salto condizionato/incondizionato
4. istruzioni di ingresso/uscita (non fornite da tutti gli assembler)

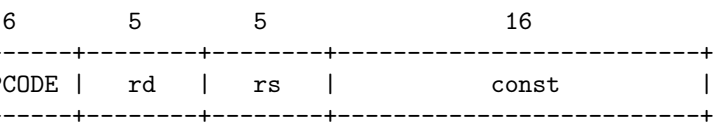
Le istruzioni sono tutte lunghe massimo 32 bit. I 6 bit più significativi si chiamano "codice operativo" (OPCODE) e indica il tipo di istruzione.

In MIPS ci sono 3 tipi di istruzioni:

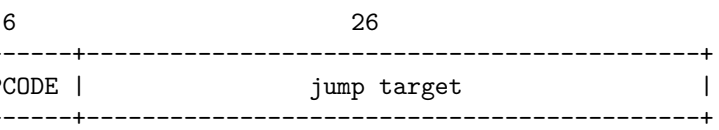
- tipo R: istruzioni aritmetico-logiche; strutturate in:



- tipo I: istruzioni di accesso alla memoria o contenenti costanti



- tipo J: salto



Sintassi:

```
istruzione field1, field2, field3
```

La lista completa di tutte le istruzioni non è riportata qui. Può essere trovata nella cartella con il materiale del corso.

Istruzioni aritmetico-logiche

istruzione	registro	registro	registro
`add`	`rd`	`rs`	`rt`
`addu`	`rd`	`rs`	`rt`
`addi`	`rd`	`rs`	`const`
`addui`	`rd`	`rs`	`const`
`sub`	`rd`	`rs`	`rt`
`subu`	`rd`	`rs`	`rt`
`subi`	`rd`	`rs`	`const`
`subui`	`rd`	`rs`	`const`
`mult`	`rs`	`rt`	-
`multu`	`rs`	`rt`	-
`div`	`rs`	`rt`	-
`divu`	`rs`	`rt`	-

Le istruzioni immediate (`*i`) hanno la peculiarità di prendere una costante al posto di un secondo registro.

Il registro di destinazione della `mult` è implicito: il risultato viene salvato in `$hi` (32 cifre più significative) e `$lo` (32 cifre meno significative). Per spostare il risultato da questi due registri viene usato `mfhi rd` e `mflo rd`. Stessa cosa vale per la divisione solo che il quoziente è posto nel registro `$lo`, mentre il resto in `$hi`.

Trasferimenti in memoria

istruzione	registro	registro
`lw`	`rd`	`const(registro base)`
`sw`	`rs`	`const(registro base)`
`la`	`rd`	`const`

La parte `const(registro base)` serve a calcolare l'indirizzo di memoria a cui si fa riferimento. L'indirizzo di memoria è pari a `const + indirizzo contenuto nel registro base`.

L'ordinamento dei byte di una parola non è da dare per scontato:

- big-endian: ordinamento da sinistra a destra

- little-endian: ordinamento da destra a sinistra

Il MIPS può operare con entrambe le modalità.

Istruzioni logiche

istruzione	registro	registro	registro
`and`	`rd`	`rs`	`rt`
`or`	`rd`	`rs`	`rt`
`nor`	`rd`	`rs`	`rt`
`sll`	`rd`	`rs`	`#bit`
`sri`	`rd`	`rs`	`#bit`

`sll` di 1 bit equivale a moltiplicare per 2. `sri` di 1 bit equivale dividere per 2.

Esistono anche le varianti immediate per `and`, `or` e `nor`.

Istruzioni di modifica del flusso Le istruzioni di modifica del flusso servono a forzare la modifica del `$pc`, rompendo il flusso sequenziale standard. Il salto condizionato viene chiamato branch, quello incondizionato jump.

In tutti i casi, il salto viene effettuato ad un offset relativo al program counter, mai ad un indirizzo assoluto.

Branch

Le istruzioni di branch hanno tutte la forma:

```
branch_condizione rs, rt, indirizzo di sato
```

Avrò quindi a disposizione un salto di massimo 2^16 byte.

istruzione	registro	registro	registro	condizione
`beq`	`rs`	`rt`	indirizzo	`rs == rt`
`bne`	`rs`	`rt`	indirizzo	`rs != rt`

Per verificare diseguaglianze usiamo le seguenti istruzioni ausiliarie. Esse caricano 1 nel registro destinazione se la condizione è avverata e 0 altrimenti.

istruzione	registro	registro	registro	condizione
`slt`	`rd`	`rs`	`rt`	`rs < rt`
`sltu`	`rd`	`rs`	`rt`	`rs < rt`

Jump

Sono possibili 3 salti assoluti:

istruzione	registro
`j`	indirizzo
`jal`	indirizzo
`jr`	`rs`

La `jal` salva `$pc + 4` nel registro `$ra` prima di saltare. Essa viene usata per implementare la chiamata a funzione. La `jr` invece viene usata per implementare il ritorno al chiamante (`jr $ra`).

Nota bene: anche i salti hanno delle restrizioni sull'indirizzo (26 bit dedicati)

Etichette Le etichette vengono usate per dare nomi simbolici a delle celle di memoria. Sarà compito dell'assemblatore tradurre le etichette in indirizzi di memoria. Sintassi:

```
etichetta: add $1, $2, $3 # anche direttiva
```