



Appunti di:

Architettura degli Elaboratori

Università di Pisa, Dip. Informatica

A.A. 2018/2019






master
COPY


master
COPY


master
COPY


master
COPY


master
COPY

Indice

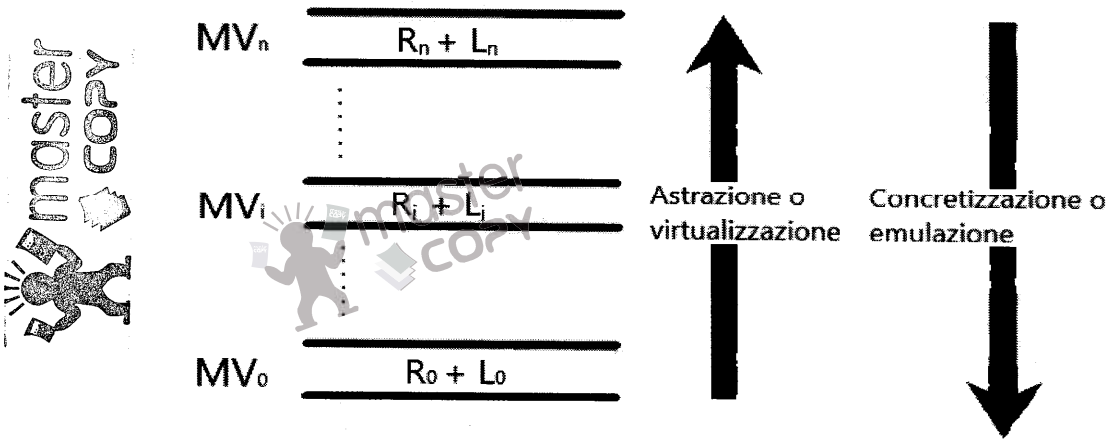
1 Macchine Virtuali....4
2 Compilazione e Interpretazione....5
3 Strutturazione verticale e orizzontale....6
4 Struttura generale di un sistema di calcolo....6
5 Ambiente globale e locale....7
6 Rappresentazione binaria delle informazioni....8
6.1 Conversione binario-decimale e viceversa....8
6.2 Numeri binari con segno....8
6.3 Notazioni di aritmetica binaria....9
7 Componenti elementari....9
7.1 Registro....9
7.2 ALU (Arithmetic Logic Unit)....10
7.3 Memoria....10
8 Algebra booleana e circuiti logici: Reti Combinatorie....11
8.1 Leggi dell'algebra booleana....12
8.2 Costruzione di un circuito logico: Sommatore....12
8.3 Costruzione di un circuito combinatorio: Commutatore....13
8.4 Costruzione di un circuito combinatorio: Selettore....13
8.5 Costruzione di un circuito combinatorio: Confrontatore....14
8.6 Reti combinatorie: Commutatore, Selettore, Confrontatore....14
9 Reti Sequenziali....15
9.1 Sintesi di una Rete Sequenziale....16
10 Implementazione di una memoria al livello firmware....16
10.1 Composizione parallela di memorie....17
10.2 Memoria Modulare Sequenziale....17
10.3 Memoria Modulare Interallcciata....18
11 Livello Firmware....18
11.1 Modello Parte Operativa-Parte Controllo (PO-PC)....18
11.2 Procedimento per la progettazione di PO e PC....19
11.3 Lunghezza del ciclo di clock....20
11.4 Ottimizzazione del microprogramma....20
11.4.1 Condizioni di Bernstein....20
11.4.2 Controllo Residuo....21
11.4.3 Microprogramma con una sola microistruzione....21
11.5 Comunicazione a Livello Firmware....22
11.5.1 Comunicazione a transizione di livello....22
11.5.2 Comunicazione a domanda e risposta....23
12 Livello Assembler....24
12.1 Compilazione e Caricamento o Loading....24
12.2 Spazio logico di indirizzamento e Memoria virtuale....25
12.3 D-Risc: un assembler didattico....26
12.3.1 Caratteristiche principali....26
12.3.2 Set d'istruzioni D-Risc....27
12.4 Interprete del set di istruzioni....28
12.4.1 Specifica della macchina firmware....28
12.4.2 Versione dell'interprete....29
13 Livello dei Processi....30
13.1 Struttura dati del supporto ai processi....30

13.2 Spazi di indirizzamento....31
13.3 Metodo a Capability....31
14 Gerarchie di memoria....31
14.1 Paginazione a domanda: principio di località e riuso....32
14.2 Insieme di Lavoro....33
14.3 Allocazione dinamica della memoria con Paginazione....33
14.4 MMU (Memory Management Unit)....34
14.5 Struttura della CPU con cache....35
14.6 Tempo di accesso alla memoria....35
14.6 Metodi di indirizzamento della cache....35
14.6.1 Diretto....35
14.6.2 Completamente associativo....36
14.6.3 Associativo su insiemi....36
14.7 Politiche di consistenza....37
14.8 Prefetching nella cache....37
14.9 Cache secondaria....38
15 Ingresso - uscita (I/O)....38
15.1 Trattamento delle interruzioni....39
16 Elaborazione in parallelo....40
16.1 Valutazione delle prestazioni di sistemi paralleli....40
16.2 Paradigma Pipeline....41
16.2.1 Pipeline con stato....41
16.3 Paradigma Farm....41
16.4 Paradigma Data-Flow....42
17 Processore D-RISC pipeline....42
17.1 Memoria e MMU....43
17.2 Processore....43
17.3 Registri generali....44
17.4 Architettura astratta e funzionamento generale....44
17.5 Operazioni aritmetiche “lunghe”....45
17.6 Ottimizzazioni delle istruzioni....45
17.6.1 Loop Unrolling....46
17.6.2 Delayed Branch....46
18 Processori Superscalari....47
18.1 Very Long Instruction Word (VLIW)....48
19 Multithreading....48

master COPY

1 Macchine Virtuali

Le funzionalità di un sistema di elaborazione possono essere viste come ripartite, secondo un certo numero di livelli, o macchine virtuali $\{MV_0, MV_1, ..., MV_n\}$, come schematizzato in figura:



Ogni livello è caratterizzato da un proprio linguaggio e quindi da una propria visione degli oggetti. La definizione di ogni livello coincide con la definizione del linguaggio in esso adottato: ciò caratterizza l'approccio language-based, ritenuto allo stesso tempo più opportuno e più rigoroso per lo studio dei sistemi di elaborazione. I livelli $\{MV_0, MV_1, \ldots, MV_n\}$ sono ordinati secondo una relazione gerarchica. Ciò significa che ogni istruzione primitiva I_k, o meccanismo del linguaggio L_i $(i > 0)$ è implementata da un programma P_k, o politica, scritta nel linguaggio L_j, dove:

- $0 \leq j < i$.
- Se I_{k1} e I_{k2} sono meccanismi distinti di L_i, questi possono essere implementati da politiche scritte rispettivamente con L_{j1} e L_{j2}, con $j1 \neq j2$.

In altre parole, il concetto che sta alla base della strutturazione a livelli è il seguente:

- Per ogni livello MV_i $(i > 0)$ l'insieme dei livelli $\{MV_j \mid 0 \leq j < i\}$ premette di implementare complessivamente il supporto a tempo di esecuzione di L_i, $STE(L_i)$, cioè una collezione di algoritmi e strutture dati che provvedono all'interpretazione dei meccanismi di L_i.
- Tale supporto è a sua volta strutturato in modo gerarchico (per $i > 1$), riconoscendo un livello più alto Mv_{i-1} il cui supporto a tempo di esecuzione è implementato dai livelli $\{MV_h \mid 0 < h < i-1)$ e così via.

Vediamo un altro modo di esprimere questi concetti:

- L'insieme degli oggetti R_i di MV_i è direttamente accessibile soltanto da parte dei meccanismi di L_i.
- A un certo livello MV_i non sono note le politiche adottate dai livelli inferiori per implementare $STE(L_i)$.

Questi principi sono alla base della strutturazione di sistemi con elevate potenzialità di: modularità, modificabilità, portabilità, manutenibilità e testabilità. Il livello MV_0 è tipicamente quello dei componenti fisici (elettronici) e a partire da questi viene costruita una *astrazione* o *virtualizzazione*