centro appunti
www.centroappunti.it
Corso Luigi Einaudi, 55 - Torino

Appunti universitari
Tesi di laurea
Cartoleria e cancelleria
Stampa file e fotocopie
Print on demand
Rilegature

NUMERO: 1103 DATA: 16/09/2014

A P P U N T I

STUDENTE: Lecce

MATERIA: Calcolatori Elettronici

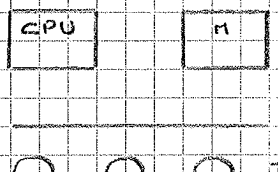
Prof. Sonza_Reorda

Il presente lavoro nasce dall'impegno dell'autore ed è distribuito in accordo con il Centro Appunti.
Tutti i diritti sono riservati. È vietata qualsiasi riproduzione, copia totale o parziale, dei contenuti inseriti nel presente volume, ivi inclusa la memorizzazione, rielaborazione, diffusione o distribuzione dei contenuti stessi mediante qualunque supporto magnetico o cartaceo, piattaforma tecnologica o rete telematica, senza previa autorizzazione scritta dell'autore.

ATTENZIONE: QUESTI APPUNTI SONO FATTI DA STUDENTI E NON SONO STATI VISIONATI DAL DOCENTE.
IL NOME DEL PROFESSORE, SERVE SOLO PER IDENTIFICARE IL CORSO.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 3 di 146

BASI DI PROGRAMMAZIONE ASSEMBLER



La CPU esegue delle operazioni ad esempio l'operazione somma, il processore ha quindi bisogno di sapere che istruzioni deve eseguire e ciò è scritto nella memoria. Nella memoria sono scritte le istruzioni sotto forma di codice. ciascuna istruzione è memorizzata in un certo numero di bit determinati da una sequenza di bit scritti dal programmatore.

Processore: deve un indirizzo di memoria, e lì può scrivere o leggere delle informazioni, alcune volte può memorizzati, possono essere memorizzate nella CPU all'interno delle Instruction register

FETCH: carica l'istruzione
EXE: esegue l'istruzione

} nella questo procedimento

Un secondo registro nella CPU è il Program counter che contiene l'indirizzo della prima istruzione, dopo aver effettuato la fetch e l'exe incrementa il Program counter (seconda istruzione)

Su un sistema special purpose il codice è tipicamente memorizzato in una Rom poiché il dispositivo eseguirà sempre quel codice ed. Mentre in un general purpose il codice sarà scritto in una RAM

(es) a = b + c; diventa una somma tra registri

Quindi l'istruzione diventa prendi il primo registro in cui è memorizzato il primo operando, prendi il secondo registro e mettila nel registro R3 dopo averli sommati

```
MOV   b, R1
MOV   c, R2
ADD   R1, R2, R3
MOV   R3, a
```

Calcolatore istruzioni alto livello → istruzioni basso livello

Assemblatore istruzioni basso livello → lasse il codice in bit corrispondenti

(es) MOV diventa 01001 ...

La MOV esegue un'operazione su una cella di memoria e un registro.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 3 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 5 di 146

. DATA direttiva che segna l'inizio di codice in cui vengono definite le variabili

? la variabile non viene inizializzata

. CODE direttiva che segna l'inizio del codice, delle istruzioni

. STARTUP
. EXIT
} l'assemblatore capisce che quello è il punto in cui inizia il programma e dove finisce quindi le due istruzioni servono per passare dal SO al programma

Struttura programma:

```
. DATA
   dichiarazione variabili
. CODE
. STARTUP
   corpo del programma
. EXIT
END
```

. model small vuol dire che il programma sarà relativamente semplice e che non supera i 64 Kbyte

esempio 2.4 Somma di due variabili

```
. MODEL SMALL
. STACK
. DATA
OPD1    DW   10
OPD2    DW   24
RESULT  DW   ?
. CODE
. STARTUP
   MOV  AX, OPD1
   ADD  AX, OPD2
   MOV  RESULT, AX
. EXIT
END
```

dichiarazione delle variabili di cui due inizializzate

calcolano i registri delle' x86

Il programma esegue 6 accessi in memoria e scrittura 1 scrittura

Registri delle' x86:

- AX
- BX
- CX
- DX

Da 16 bit ciascuno

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 5 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 7 di 146

L'operazione ADD AX, VET+2 contiene due somme, esse però vengono fatte da due parti diverse, VET+2 viene fatta dall'assemblatore mentre AX+(VET+2) viene fatta dal Processore

Quando il vettore è molto grande non posso scrivere troppe istruzioni!

Devo calcolare l'indirizzo delle celle in via separata:

ADD AX, [VET DI] → istruzione chiamata modo di indirizzazione

Esistono due ulteriori registri che permettono di costruire gli indirizzi, che sono SI e DI su 16 bit, posso mettere utilizzare anche BX

NB • VET[DI] non vuol dire prendi l'elemento di VET in posizione di DI ma vuol dire prendi l'indirizzo di VET e sommaci il valore che in quel momento ha DI

• ADD DI, 2 incremento DI di 2

• Gestione dei cicli:

CX contiene il numero di volte che bisogna ripetere le operazioni

Esiste un'istruzione chiamata JUMP (JNZ) che permette di "saltare" delle operazioni è come un GOTO (dell'undr.)

In assembler non esiste l'IF ma si realizza in due istruzioni: verifica una condizione, se risultato è uno, salta

```
DEC  DX
CMP  CX,0
JNZ  etab

for (i=0; i<50; i++)
    {_____}

MOV  CX,50
etab: _____        etichetta: dove c'è l'istruzione
                              che deve eseguire
      _____
DEC  CX
CMP  CX,0
JNZ  etab
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 7 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 9 di 146

S e z sono i due flag a seconda del risultato dell'operaz. assumono un valore che è quello riportato in tabella (a seconda che il risultato sia positivo/negativo/non)

Alcune istruzioni (ADD) quando devono essere eseguite settano i flag.

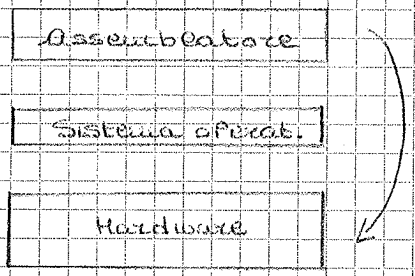
- La JNZ del programma esegue un test, se z è a 0 esegue il salto se il flag è a 1 non esegue il salto e si esegue l'istruzione succ. in memoria cioè mov result ... Quando esegue il salto, il computatore va dove viene indirizzato cioè dove dice la JNZ in questo caso fab indica un'istruzione precedente e allora va all'indietro ed esegue un ciclo. Tutto ciò tramite l'etichetta fab che contiene l'indirizzo dell'istruzione a cui saltare

esempio 4:

```
DIM     EQU   20
        .MODEL  small
        .STACK
        .DATA
VETT    DB    DIM DUP(?)
        .CODE
        .STARTUP
        MOV   CX,DIM
        MOV   DI,0
        MOV   AH,1
```

→ continua

Obiettivo del programma è essere dei caratteri memorizzarli in un vettore e visualizzarli al contrario



interagisce con l'hardware attraverso il sistema operativo! Quindi è lui che lavora

Le istruzioni MOV AH,1 INT 21H serve per fare input cioè servono a chiamare il sistema operativo e a dettare il carattere premuto su tastiera per metterlo (il suo codice ASCII) nella variabile AL.

(INPUT)

Teniamo presente che AX su 16 bit è fatto da AL e AH entrambi da 8 bit, che rappresentano rispettivamente parte alta e bassa del registro AX

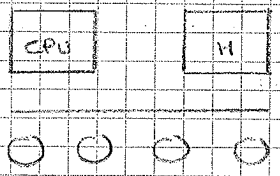
© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 9 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 11 di 146

Architettura - organizzazione vedi 8086

ISA = istruzioni che il produttore fornisce del processore affinché di possano costruire

- Come il processore si interfaccia con la memoria?

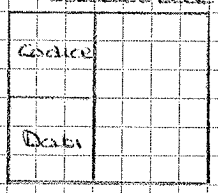


L'8086 era in grado di fornire indirizzi su 20 bit cioè vuol dire che i segnali mandati dalla CPU sono su 20 bit

Su 20bit posso rappresentare 2^{20} indirizzi, cioè può vedere 1Mb di memoria (pochissimo) ma tanto per i tempi.

Considerando i registri DI e SI posso scrivere MOV AX, [SI] cioè sposta 16 bit dalla memoria ad AX, e secondo operando indica una locazione di memoria, in particolare quella il cui indirizzo sta in SI

Problema: indirizzi processori su 20 bit e registri su 16 bit come faccio a sestarli?



Per risolvere il problema i costruttori introdussero due registri accanto alla CPU ovvero CS e DS che contengono gli indirizzi di partenza di codice e dati (che hanno sei ultimi 4 bit a zero) ma solo i primi 16 bit

OFFSET: distanza in byte dell'istruzione che vuoi eseguire alla prima (cui che viene memorizzato nel PC)

Per accedere alla memoria prende il primo indirizzo della mia contenuto in CS e ci somma quello contenuto nel PC e ottiene uno su 20 bit con cui può accedere alla memoria (nella somma tengo conto degli ultimi 4 bit mancanti a zero)

(IP) in ambiente intel è il program counter

- Rappresentazione delle parole

MOV VAR,AX scrive in VAR quello che è scritto in AX ma in che ordine ci si mette? Little endian? Big endian? Dipende dal processore. Little endian vuol dire indirizzo minore ci va le meno significativo!

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 11 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 13 di 146

è un operatore! da + (somma) è un operatore e vengono ridotte ad una costante non vengono fatte le tempo di esecuzione pertanto queste operazioni devono diventare costanti al tempo di assemblaggio pertanto mov BX, AX+2 non è errata.

(2) Modi di indirizzamento

specifica l'operando di una istruzione (1, 2, ... o nessuno)

mov (AX), (BX) 2 operandi

dec cx 1 operando

I modi di indirizzamento possono essere registri, possono stare in memoria o nell'istruzione stessa (costante)

mov VETTORE, AX è una cella di memoria che deve essere raggiunta tramite somma di indirizzo + offset

I modi di indirizzamento sono 7:

1) modo registro addressing mov BX, AX

2) modo immediate " mov BH, 07h

Solo il secondo può essere una costante! La costante viene scritta sul codice macchina o su 8 o su 16 bit l'assemblatore decide come rappresentarlo in modo da non sprecare memoria. Il processore legge la costante in completamento a due, ma il programmatore può scrivere in decimale, esadecimale ecc...

Però se è esadecimale alla fine devo mettere 0...h. Per parlare capire che è esadecimale.

mov cx, 0FFFFh esadecimale

mov cx, 0110011b binaria

mov cx, 30 senza specificare è decimale

3) direct addressing mov AX, TABLE come variabile

mov AX, TABLE+2 offset di table + 2 } solo usuali

mov AX, TABLE[2] offset di table + 2 }

mov AX, TABLE-2

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 13 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 15 di 146

5) Base relative addressing

MOV AX, [BX]+4 — accediamo alla cella di memoria con offset contenuto in BX e somma 4

è uguale a scrivere MOV AX, 4[BX] oppure MOV AX,[BX+4]

6) Direct indexed addressing

utilizzato per l'accesso al vettore

MOV AX, TABLE[DI] — prendere l'offset di table sommarlo a DI e accedere

esempio:

```
for (i=0; i<DIM; i++)
    vett[i]=0;

    MOV  DI,0
    MOV  CX,DIM
e3: MOV  VETT[DI],0
    ADD  DI,2
    DEC  CX
    CMP  CX,0
    JMP  e3
```

posso usare SI, DI, BX

non c'è bisogno di usare PTR perché è tutto stato dichiarato come DW

7) Base indexed addressing

MOV AX, TAB[BX][DI]+6

si combinano due registri e una costante, somma i registri, somma la costante e tutto si somma a DS e si accede spostando in AX

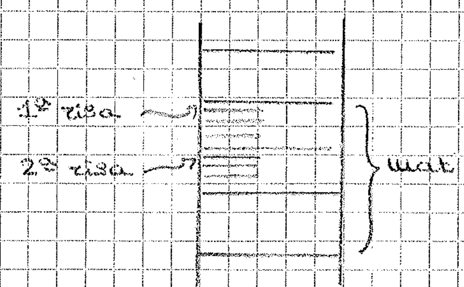
A cosa serve? Nelle matrici!

L'x86 non supporta le matrici né le struct, allora come le si fa? esempio:

```
int mat[K1][K2];
;
mat[i][j]=0;
```

} in C!

In C si alloca K1*K2*2 byte in verticale (ricorda come è organizzata la memoria)



Primo elemento della riga i-esima
offset (i,j) = i* 2K1 + 2j

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 15 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 17 di 146

Istruzioni

L'assembler x86 non è ortogonale! Cioè questa proprietà per cui dato un qualunque operando possiamo utilizzare qualsiasi modo di indirizzamento

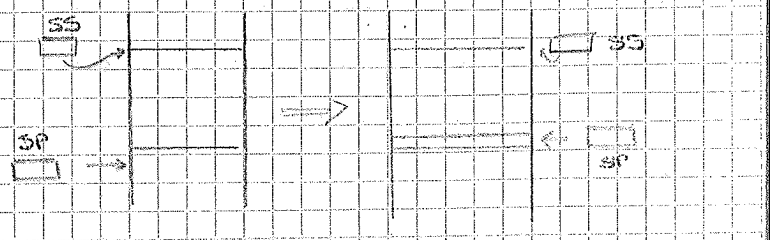
es. mov [ ], [ ] non posso mettere nei due blocchetti due celle di memoria questo perché il processore non lo supporta

Le istruzioni assembler sono raggruppabili in varie categorie, ad esempio:

- Trasferimento dati
- Aritmetiche
- Manipolazione di bit
- Controllo di flusso

• Istruzioni di trasferimento dati

- L'istruzione più importante è la mov con tutte le caratteristiche già viste (mov destinazione, sorgente) non possono essere due celle di memoria, ne due variabili, i due operandi devono avere la stessa dimensione ecc...
  Ci sono innumerevoli regole per la mov! (vedi sedes)
- Altre istruzioni sono ad esempio xchg che supporta due registri come operandi, e scambia i due registri, questa istruzione non è indispensabile potrebbe essere realizzata con più mov
- xlat non ha operandi! Opera su registri stabiliti
- LEA load effective address (offset) ha due operandi, registro e variabile prende l'offset della variabile e lo mette nel registro! È un'operazione importantissima che viene usata spesso è equivalente a mov registro, offset variabile, Uso più importante copia di un vettore di interi
- PUSH e POP il sistema operativo quando interviene nella computazione del processore utilizza una zona detta stack, in particolare utilizza ds e ss, e l'ultima è una zona di memoria di tipo lifo. Attraverso lo SP (stack pointer) che contiene l'offset all'interno del segmento di stack dell'ultimo elemento inserito, si effettuano operazioni di PUSH e POP

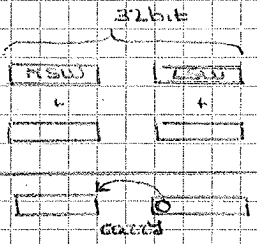


all'inizio lo SP punta nulla! Perché lo stack è vuoto, quando lo riempio andrà ad inserire verso decrementando

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 17 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 19 di 146

- INC e DEC incrementano di uno e decrementano di uno, possono essere definite su variabili (celle di memoria) o su registri.
- ADC: Estendere le operazioni su 32bit



Quando ad addizionare le LSW devo tener conto se c'è o meno riporto! Quindi la seconda ADD deve aggiungere 1 se il flag del riporto è settato ad 1 per far ciò è stata introdotta la ADC

esempio: l'operando 1 contenuto in AX e BX con AX contenente la MSW e BX la LSW mentre l'operando 2 sta in CX:DX il risultato si troverà in CX:AX

```
ADD DX, BX
ADC CX, AX
```

Esiste un'analoga istruzione SBB per effettuare una sottrazione su 32bit

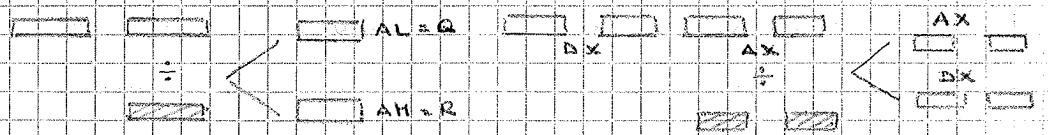
- MUL e DIV servono per moltiplicazioni e divisioni senza segno! Se ho bisogno di effettuare l'operazione con numeri con il segno devo usare le corrispondenti IMUL e IDIV

es. MUL BX vuol dire AX * AX → DX:AX

MUL BL vuol dire BL * AL → AL

DX viene usato come registro aggiuntivo nel caso di riporti oppure operazioni che vogliono su tot bit. L'operando può essere una cella di memoria, una bisogna specificare su quanti bit si trova la variabile stessa

es. DIV operando con operando = divisore



AX = dividendo di operazioni su 16 bit

DX alta AX bassa = dividendi di operazioni su 32bit

Se il quoziente non è rappresentabile il processore si ferma e chiama una procedura che segnala l'errore!

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 19 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 21 di 146

Supponiamo di avere l'istruzione JE label, essa verifica se flag di zero è uguale a uno e modifica quindi l'indirizzo del PC, come? Come prende l'indirizzo? Lo deduce dalla posizione di label nel codice, conta quanto dista label dall'istruzione corrente, e ottiene l'offset, ne deduciamo che siccome questa distanza deve essere memorizzata in 1 byte massimo (da -128 a 127) non posso saltare troppe righe in qua? Allora come faccio se devo saltare molto più in là? Uso una JMP che invece salta più dove voglio. Esiste una JMP FAR che sta su 5 byte invece che su 3 byte, e contiene cs e IP.

Osservazione: Esistono in assembler 3 tipi di salti:

- far cs e IP
- near IP
- short $\pm$ (distanza dall'istruzione corrente)

- Istruzioni di iterazione

- LOOP permette di eseguire un numero finito di volte un certo codice dipende dal contatore CX quando CX viene decrementato fino a zero termina anche LOOP.
- LOOPE

- Istruzioni per la manipolazione dei bit

- AND dest, sorg esegue l'and logico bit a bit tra il contenuto di dest e quello di sorg il risultato è messo in dest

  es. $y_1\ y_2\ y_3\ y_4\ \dots\ x_1\ x_2\ x_3\ x_4 \longrightarrow 0\ 0\ 0\ 0\ x_1\ x_2\ x_3\ x_4$

  ho trasformato i 4 bit alti in zeri.

  Questa istruzione ha gli stessi effetti della ADD cioè ad esempio dest e sorg non possono essere due registri.
- OR dest, sorg esegue l'OR bit a bit quindi effettivamente fa l'operazione opposta dell'AND per esempio se utilizzo AND per convertire da codice ASCII a binario utilizzerò invece l'OR per convertire da binario a codice ASCII (numero su 8 bit)

  es. OR AL, 30H usabile a ADD AH, 48 oppure OR AL, '0'

  AND AL, 0FH usabile a SUB AL, 'a'
- XOR esegue l'XOR bit a bit tra dest e sorg XOR DX, DX (mette a zero)
- NOT operando esegue il complemento bit a bit dell'operando questa istruzione non va confusa con NEG (cambia il segno)

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 21 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 23 di 146

Nella parte che riguarda questa mettiamo degli zeri

N.B SHR AL,2 equivale a dividere per 4 } con operandi
SHL AL,3 equivale a moltiplicare per 8 } sempre uguale

È un errore grave usare XOR e AND quando posso usare queste che impiegano molti cicli di clock in meno, in AL ottengo il risultato, il resto viene conservato solo se è 1

- SAL operando, contatore (identica alla SHL in tutto e per tutto)
- SAR operando, contatore shifta e rimette l'ultimo bit VCHE dovrebbe buttare nei posti vacanti dell'inizio
- ROL operando, contatore (altre ROR, RCL, RCR)

rimettono i bit che eliminano nei posti vacanti dell'inizio.

esempio: effettuare le rotate in un byte

scambio i bit a a

Formato delle istruzioni macchina

È complicato trasformare il linguaggio assembler x86 in linguaggio macchina. Le istruzioni macchina dei 8086 hanno una dimensione da 1 a 6 byte dove alcuni di questi bit contengono il codice operativo (l'istruzione) e una parte di essi che contengono gli operandi (attenzione: alcune istruzioni non contengono gli operandi e hanno solo codice operativo)

Tempo di esecuzione

Nel manuale di un processore possiamo andare a vedere quanto tempo richiede una certa istruzione, il compilatore che converte in linguaggio macchina studia quali sono i comandi più rapidi che può utilizzare. Il tempo dipende:

- numero dei cicli di clock (cambia a seconda del processore ecc)
- tipo di istruzione
- posizione degli operandi (registri, memoria)
- allineamento degli operandi in memoria

Come si calcola il tempo di esecuzione?

- Considerato l'effettivo tempo di accesso EA
- tempo di accesso all'operando
- tempo di esecuzione dell'istruzione

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 23 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 25 di 146

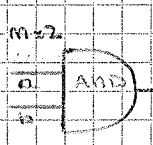
Sistemi combinatori e sequenziali

Un sistema può essere:

- combinatorio: le uscite dipendono solo dagli ingressi in quel momento, non dalla storia passata
- sequenziale: dipende dagli ingressi correnti e passati, es. memoria

(1) Il comportamento di un sist. comb. può essere descritto, a livello di specifiche, da:

- una tavola della verità = ∀ combinazione di ingresso → combinazione in uscita o valore d'uscita



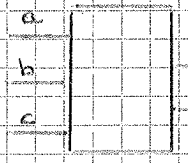
u = a ∧ b

0 and 0 → 0
0 and 1 → 0
1 and 0 → 0
1 and 1 → 1

Difetto: m tende a essere esponenziale con l'aumentare del numero di bit d'ingresso

- funzione booleana = ottenuto il valore dell'uscita inserendo i dati in ingresso.

esempio: codificatore prioritario



funzione booleana

$x_1 = a + b$

$x_0 = a + b\bar{c}$

tavola della verità

a	b	c	x0	x1
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	0

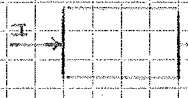
eccetera...

la funzione booleana descrive in maniera più compatta le stesse info della tabella

(2) Il comportamento di un sistema sequenziale è definito attraverso:

- valori in correnti
- variabili di stato (rappresentano la storia passata)

esempio:



il modello osserva i valori ogni cot di tempo se trova una sequenza di 101 ottiene all'uscita pari a 1.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 25 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 27 di 146

Porte logiche

Attraverso l'uso dei transistor è possibile implementare una qualsiasi porta logica:

AND, NAND
OR, NOR
XOR, XNOR
} + NOT

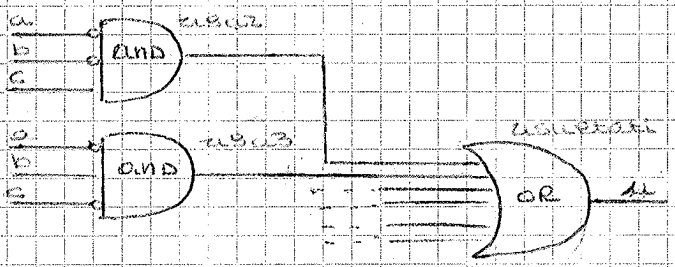
Questi componenti devono essere assemblati, combinati in maniera opportuna per formare le transistor

esempio:

1)	0	0	0	0
2)	0	0	1	1
3)	0	1	0	1
4)	0	1	1	0
5)	1	0	0	1
6)	1	0	1	1
7)	1	1	0	1
8)	1	1	1	1

Ho 6 uscite a 1
e 2 a 0

Per rappresentare le uscite ad uno posso usare porte and combinate con or
Per esempio la riga 2 diventerebbe



Ma facendo in tal modo non si ottiene un circuito ottimizzato, è una forma semplice ma non fornisce la soluzione migliore, tralasciamo trattare un altro metodo migliore.

Legge per un circuito ben formato

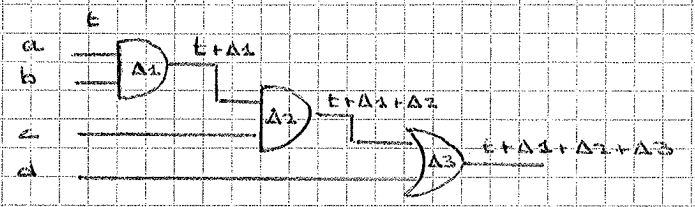
1) Ciascuna linea e ciascuna porta è un CCBF
2) Se collego (giustapponendo) due CCBF ottengo un CCBF
3) C1 e C2 sono due CCBF il circuito ottenuto connettendo le linee d'uscita di C1 con le linee d'ingresso di C2 ottengo un CCBF
4) Se n1 e n2 sono due ingressi di un CCBC e li connetto ottengo un CCBF
5) non devono esserci cicli!

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 27 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 29 di 146

Velocità del circuito

Associata a ciascuna porta logica è associato un ritardo cioè il tempo (in nanosecondi) che la porta impiega a commutare da 0 a 1 o viceversa. Avendo tutte porte logiche diverse con le proprio ritardo, il ritardo complessivo sarà la somma di tutti i ritardi.



Per essere il valore esatto associato all'ultima porta, deve far passare un tempo sufficiente che bisogna calcolare! Dobbiamo sapere che sia finita la transizione (delle porte) cioè fin quando la porta non si è stabilizzata, non posso apprezzare un nuovo valore di uscita.

Cammino critico

Rappresenta il cammino più lungo che collega l'ingresso con l'uscita cioè bisogna trovare quale tra i vari è il ritardo massimo. (Misurazione su stato del cammino più lungo)

Questo cammino è individuato quando si calcola la profondità del circuito

Circuiti sequenziali

Supplementano funzioni dipendenti dal tempo, memorizzano info e sfruttano i ritardi delle porte.

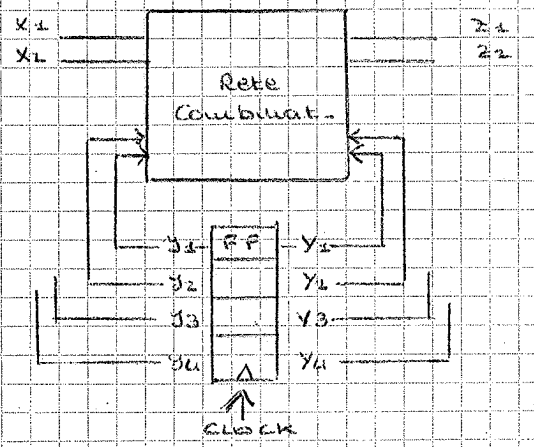
Devono essere in grado di memorizzare lo stato precedente (es attraverso una variabile di stato) allora abbiamo bisogno di memorizzare con un modulo dei bit. Il modulo che fa ciò è chiamato FLIP FLOP.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 29 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 31 di 146

Un circuito è detto sincrono quando è dotato di un clock e di flip-flop che permettono al circuito di assumere valori che dipendano anche dagli stati precedenti.

- Modello di Huffman



I valori dei flip-flop vengono memorizzati in una tabella di stato esattamente equivalente al diagramma degli stati

esempio: Se la variabile di stato deve assumere 5 valori le numero di FF è di 3 perché con 3 posso rappresentare 2^3 stati mentre con 2 FF solo $2^2 = 4$

In generale dati n numero di stati ci posso rappresentare con $\log_2 n$ (prendendo l'intero superiore)

Passaggi: (per ottenere il modi di Huff)

1) Guardare il num degli stati e a seconda di questo determinare il numero di Flip-Flop (Minimizzazione)
2) Assegno a ciascun stato la corrispondente codifica dei flip-flop

Al questo punto posso cambiare la tabella degli stati nella corrispondente tabella dei flip-flop

es: A 0 B 0 ⟶ 0 0 0 0 0 0 0

A 1 B 0 ⟶ 0 0 0 1 0 0 1 0

3) Prendere la tab degli stati e trasformarla in quella dei FF
4) Ottengo i flip-flop con la solita combinatoria

⟹ Ottengo il circuito combinatorio equivalente

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 31 di 146

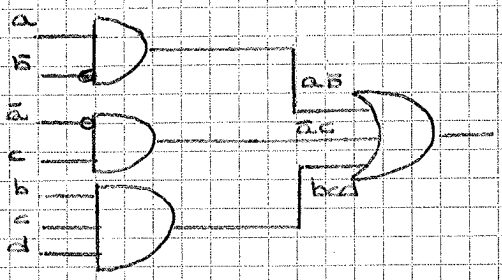
© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 33 di 146

Funzione booleane

- costruire la tavola della verità
- Trovare il corrispondente circuito minimo

• $f = a \cdot \bar{b} + \bar{a} \cdot c + b \cdot c \cdot d$

a	b	c	d	f
0	0	0	0	0
1	0	0	0	1
0	1	0	0	0
0	0	1	0	1
0	0	0	1	0
1	1	0	0	0
1	0	1	0	1
1	0	0	1	1
0	1	1	0	1
0	1	0	1	0
0	0	1	1	1
1	1	1	0	0
1	0	1	1	1
1	1	0	1	0
0	1	1	1	1
1	1	1	1	1



circuito non minimo

cd \ ab	00	01	11	10
00	0	0	0	1
01	0	0	0	1
11	1	1	1	1
10	1	1	0	1

mappa di Karnaugh

cd 1° termine

3 cubi

1° termine āc

ab 3° termine

$f = \bar{a}c + cd + ab$

1 termine 2 termine 3 termine

funzione minimizzata

Progetto di sistema sequenziale

$x_1 = 1 \Rightarrow$ Inserita moneta da 15

$x_2 = 1 \Rightarrow$ Inserita moneta da 10

$z = 1 \Rightarrow$ Inserito almeno 30

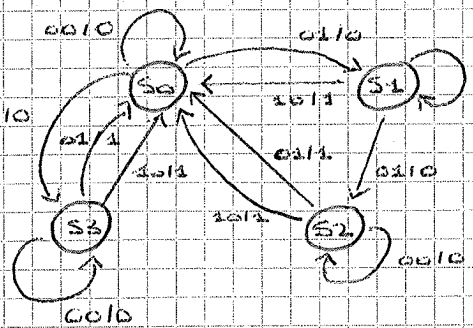


diagramma degli stati

stati: monete che possiamo inserire per avere almeno 30 cent

S0 = 25 + 10

S1 = 25 + 25

S2 = 10 + 10 + 10

S3 = 10 + 10 + 25

è l'ordine di inserimento non importa

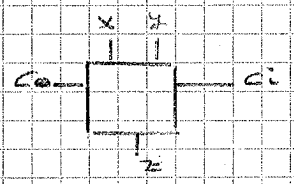
© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 33 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 35 di 146

- Moduli aritmetici

A questa famiglia di moduli corrisponde ad esempio il Sommatore
Possono avere complessità variabile a seconda di:
- Tipi di dati supportati
- tipi di operazioni
- velocità

Sommatore seriale
costa pochissimo
Supporta la presenza di un bit in più e un carry



È formato da un FAP-FEP e un Full-adder
Possiamo utilizzare un carry-lookahead (costoso ma veloce) oppure un ripple carry adder (economico ma lento) oppure delle combinazioni dei due.

ALU

Si occupa delle operazioni aritmetiche e logiche. La ALU gestisce quindi operazioni del tipo combinatorio. È fatta da segnali in input, in output e alcuni di controllo, può avere inoltre un carry-in e un carry-out

Comparatore a n bit

Riceve dei segnali in input li computa e stabilisce se sono uguali, se uno è sono entrambi quale è il maggiore. Può avere un segnale di ENABLE. Per valutare due numeri passo confrontare ogni n bit. Se i primi n sono diversi allora subito posso decidere quale è il più grande se sono uguali guardo ai successivi bit seguenti ecc ecc.. In tal modo avrò un comparatore per i primi n bit altri per i seguenti quest'ultimi vengono attivati (a seconda del valore dell'enable) solo se necessario.

- Registro a m bit

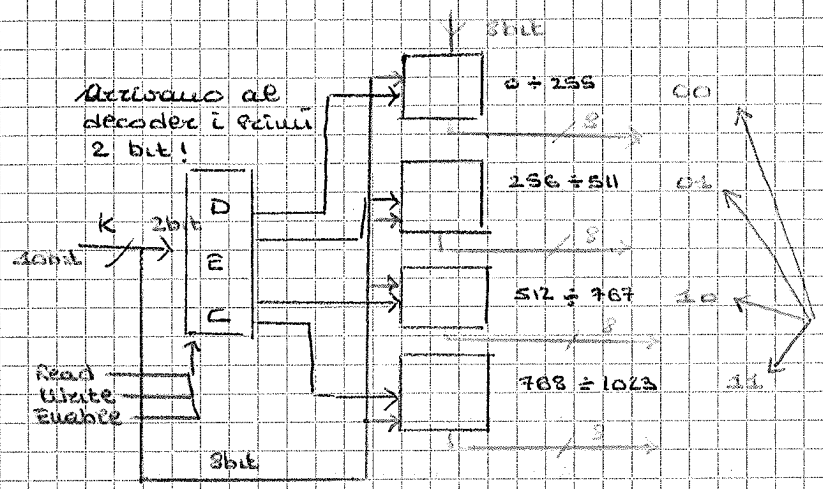
Passiamo a moduli di tipo sequenziale. I registri a m bit sono in grado di memorizzare m bit, essi sono costituiti da tanti FAP-FEP.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 35 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 37 di 146

Esempio: Disponho di moduli da 8 bit e ho un numero di parole $N = 1024$ ($m=10$) da memorizzare, ciascun modulo però gestisce solo $N = 256$ parole come realizzo la memoria?

Soluzione: Ho bisogno di 4 moduli da $N = 256$ parole per arrivare a 1024. Posso pensare di dividere le N parole nei 4 moduli.



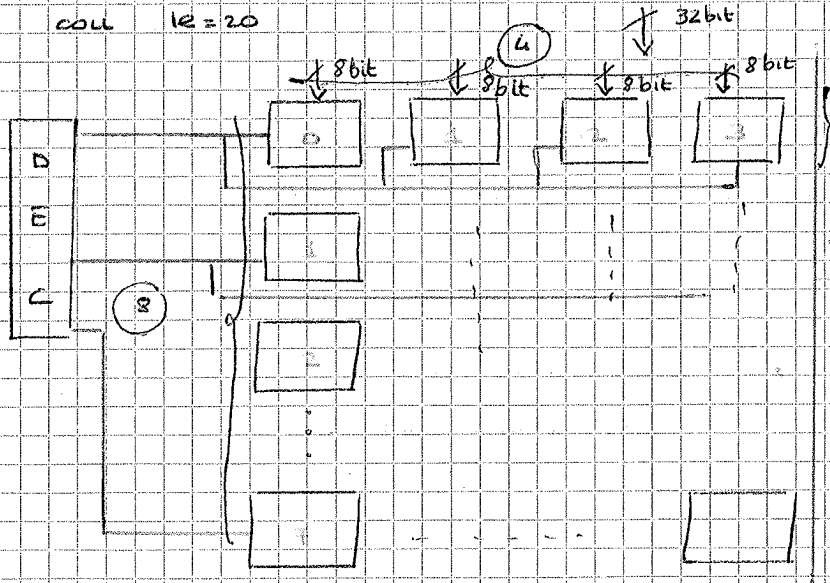
A seconda di che "parola" mi serve devo attivare uno dei 4 moduli, come faccio a capire quale modulo? Guardo i primi 2 bit).

Gli altri 8 bit passano attraverso un altro collegamento all'unico modulo attivo (poiché grazie ai 2 bit è stato attivato dall'enable) e vengono quindi utilizzati.

In uscita avrò un bus (linee verdi) che a seconda dei segnale attivo (RD, WR, EN) si comporta in un certo modo.

Esempio: Jolelo costruire una memoria con $m = 32$ bit e $N = 2^{20}$ bit avendo a disposizione dei moduli a $m = 8$ bit e $N = 128$ K

Soluzione: 8 bit di indirizzo dei moduli sono $k = 17$ devo costruire una con $le = 20$



3 bit più significativi vanno al decoder che attiva uno solo dei moduli da 4 blocchi, il resto dei 17 bit (meno significativi) vanno in parallelo ai 4 moduli.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 37 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 39 di 146

Sequenza di istruzioni:

- PC ⟶ AR
- AR ⟶ ABUS
- invio segnali di controllo alla memoria
- DBUS ⟶ DR
- DR ⟶ IR
- aggiornamento del PC

} Fetch

alla fine PC contiene l'indirizzo dell'istruzione seguente mentre l'IR contiene il codice dell'istruzione da eseguire

Se tutto viene gestito dall'unità di controllo

Usare l'accumulatore significa memorizzati i risultati delle istruzioni esso impatti è un registro.

Estensioni:

Rispetto al processore minimo, l'8086 ha a disposizione più AR, questi registri possono avere diversi utilizzi, essere specifici per un certo compito. Il numero di registri di un processore è fondamentale.

Alcune differenze risiedono anche nel tipo di istruzioni che i moduli possono eseguire (ad esempio se la ALU può eseguire solo somme o no)

Il registro di stato contiene un blocco di bit di stato e uno di controllo (es. bit di interrupt, bit di trap). Ci potrebbe essere ~~Presenza dello~~ stack implementato da stack register e stack pointer.

Lab 03.2

$c = (A \times 2) + (A \text{ xor } B)$

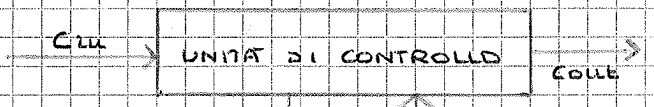
© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 39 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 41 di 146

- Unità di controllo

Processori:

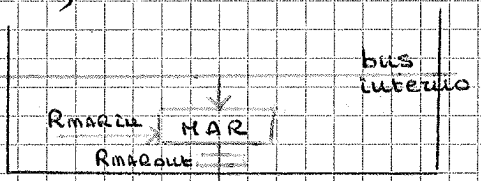
- Unità di elaborazione: strutture hardware con i registri
  ALU
  strutture di supporto (multiplexer...)
- Unità di controllo: deve pilotare l'unità di elaborazione, attivando i segnali che vanno ai vari moduli sulla base di segnali che provengono dai moduli stessi o dall'esterno.



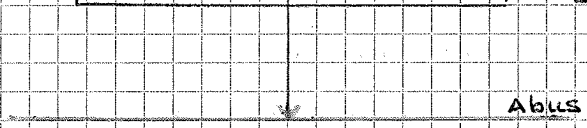
I segnali possono essere anche dei "flag" settati a 1 o a 0 a seconda di cosa bisogna fare.

I segnali inoltre devono essere caricati sul BUS uno e uno solo alla volta, pertanto si decide che valori devono andare ai registri, in Rin e Rout.

MAR

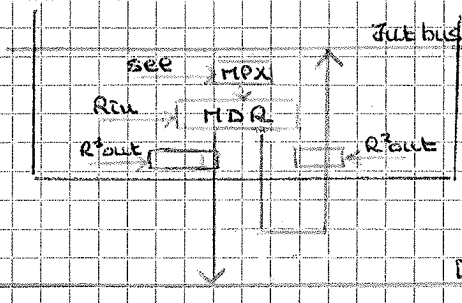


Serve a contenere il valore che il processore vuole caricare sull'Abus



Il segnale RMARout scrive solo sul bus esterno cioè l'ABus

MDR



È più complicato perché può essere da tutti e due i bus e scrivere su entrambi

Attraverso R^1out e R^2out l'UC può far capire su che bus bisogna scrivere.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 41 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 43 di 146

MOV Ru, R5

Ho fatto il fetch dell'istruzione che ora si trova sull'IR e l'UC la legge e deve decidere cosa fare, trattandosi di una MOV essa:

R5 out	carica R5 sul bus esterno
Ru in	attiva il bus interno di Ru e carica il valore che legge (quello di R5)

ADD R1, R2, R3

Voglio sommare R2 e R3 e spostarli in R1:

R2out	carico il valore di R2 sul bus esterno
Yin	lo appoggio momentaneamente su Y
R3out	carico R3 sul bus interno della ALU
Carry in = 0	l'UC pone il carry in a zero perché non voglio riporti
ADD	attivo la ALU che somma
Zin	
Zout	carichiamo sul bus interno
R1 in	R1 prende ciò che c'è sul bus interno

MOV Ru, S

Voglio spostare nel registro Ru il valore immediato S, memorizzato nel codice macchina, che a sua volta si trova nell'IR. Quindi si accede all'istruzione register. In particolare a quella parte destinata a contenere il codice macchina di S.

IRout
Ru in

MOV R3, var

Il processo è simile a quello precedente, tranne per il fatto che, quando si accede all'IR, questo contiene l'indirizzo in memoria di var. Si accede quindi alla memoria e si prende il valore di var.

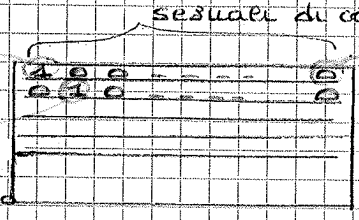
IRout	... R3 in
MARin	• Osservazione:
MAR out	Prima di queste istruzioni va
READ	eseguito il fetch
aspetta MFC	
SEL	
MDRin	
MDR out	...

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 43 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 45 di 146

Più grande ed è quindi più costosa.

In questo progetto avremo una sorta di "mappa" che per ogni segnale di controllo e per ogni istante di clock ci dice cosa è attivo e cosa non lo è



se mi accorgo che un valore non è corretto basta cambiare solo quel valore, pertanto ha un'elevata flessibilità.

- Le Procedure

Attraverso le Procedure è possibile scrivere una sola volta delle istruzioni che devono essere ripetute più volte

Vantaggi saranno: maggiore flessibilità del codice, risparmio di tempo e risparmio di memoria.

Ottimizzazione: per evitare di raffreddare le prestazioni (causa l'accesso in memoria) posso trasformare le Procedure in macro.

Definizione di Procedura:

```
label    PROC
         ------
         ------   } codice della Procedura
         ------
label    ENDP
         ------
         ------
         CALL label
```

Per chiamare la routine uso l'istruzione CALL target, questa va a mettere nello stack l'indirizzo del target ovvero mette l'IP (l'istruzione successiva alla CALL), poi la CALL esegue un salto alla prima istruzione della Procedura.

Istruzione RET

Esegue una POP dallo stack prelevando l'indirizzo di ritorno.

E salta a tale indirizzo.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 45 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 47 di 146

In assembler posso passare i parametri attraverso variabili

(1) globali (sconsigliato)

```
LUNG  EQU 100                    SOM_VET  PROC
                                          PUSH SI
      .MODEL small                        PUSH AX
      .STACK                              PUSH CX
      .DATA                               MOV  SI,0
                                          MOV  AX,0
VET   DW  LUNG DUP(?)                     MOV  CX,LUNG
SOM   DW  ?                       ciclo:  ADD  AX,VET[SI]
      ----                                ADD  SI,2
      .CODE                               LOOP ciclo
      ----                                MOV  SOM,AX
      CALL SOM_VET                        POP  CX
                                          POP  AX
                                          POP  SI
                                          RET
                                  SOM_VET ENDP
```

Tale metodo è fortemente sconsigliato perché adatto ad utilizzare le vettore e parametri quali sum e som.

Un' ulteriore metodo implementa l'utilizzo dei registri. (2)

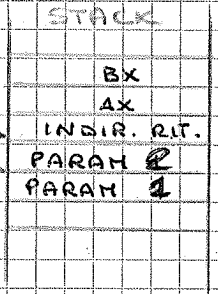
```
      .CODE                              SOM_VET  PROC
      ----                                        PUSH BX
      MOV AX, LENGTH VET  } equivalenti ad        PUSH CX *
      LEA BX,VET          } usare variabili       MOV  CX,AX
      CALL SOM_VET        } globali               !
      MOV SOMMA, AX                               !
                                                  POP  CX
                                                  POP  BX
```

* Non viene fatta la PUSH di AX perché quest'ultimo è il valore di ritorno della funzione! Non va quindi riportato al valore di prima!

Svantaggio: pochi registri a disposizione

(3) Uso dello stack:

```
PUSH 1
PUSH 2
CALL P1
```




```
(P1) PROC
     PUSH AX
     PUSH BX
     ---
     ---
```

PRIMA ISTRUZIONE
MOV BP,SP
in tal modo sarà il fondo dello stack
E per accedervi utilizzerò [BP+2]

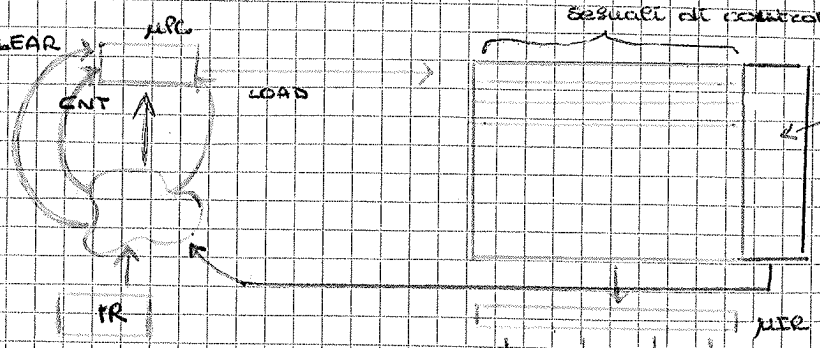
Come prendo i due parametri che sono sotto l'indirizzo di ritorno?

Attraverso il registro BP che viene utilizzato per accedere allo stack!

Lo stesso meccanismo può essere utilizzato per memorizzare il parametro di output. Posso metterlo o sotto i parametri 1 e 2 oppure tra 1 e l'indirizzo di ritorno.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 47 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 49 di 146



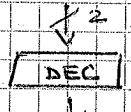
P(IR) ⟶ address ↑ istruction register

esiste una funzione combinatoria che associa al valore di IR un indirizzo

Se maggior parte delle microprogrammate è le costo. Tuttavia molto spesso accade che i segnali di controllo sono troppi, si vede allora che alcuni di questi bit non assumono tutti i possibili valori cioè 2^N ma valori solo un tot.

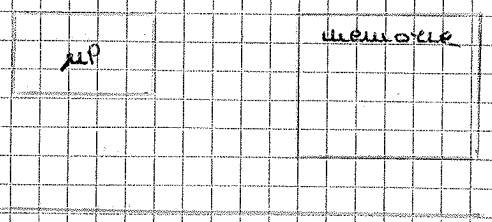
esempio: N = 4 → 1000, 0100, 0010, 0001 → memoria con N = 2 bit → 00, 01, 10, 11

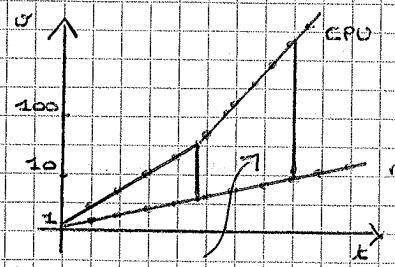
è interposto tra la memoria e le μIR un decoder



Questa codifica è possibile perché il numero di bit attivi è sempre solo 1 (compatibilità dei segnali)

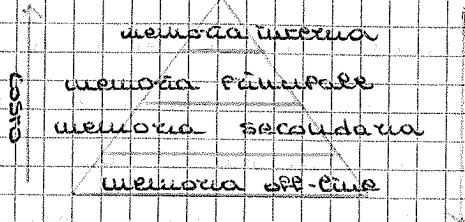
- Introduzione alle memorie





La differenza di velocità nel tempo tra processori e memorie aumenta

Bisogna gestire questa differenza perché i processori creati diventano sempre più veloci mentre la velocità della memoria resta più o meno la stessa



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 49 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 51 di 146

trasferimento indica ea velocità con ea quale vengono trasferiti tali blocchi. Si misura in tempo/byte

3) modi di accesso

- casuale: ogni unità ha un indirizzo specifico e ee dato possono essere lette in qualsiasi ordine. inoltre viene garantito un tempo di accesso costante per tutte ee celle a cui accedo
  RAM: ROM (cancello) memorie a semiconduttore
- sequenziale: ee celle possono essere lette solo da un certo verso quindi solo secondo un ordine prefissato. Nastri
- diretto: ogni blocco ha un indirizzo e la ricerca del blocco avviene in maniera sequenziale. dischi magnetici
- associativo: l'accesso avviene attraverso confronto del contenuto della cella. con quello specificato in una maschera
  content addressable memories (CAM)

4) DURevolezza del contenuto

In questo tipo di memorie si legge il contenuto e si cancella il contenuto della cella stessa. Oppure alcune memorie perdono il loro contenuto se esse non vengono accedute dopo un tot di tempo.
Volatilità: Un esempio è la memoria RAM, essa perde il proprio contenuto quando non è alimentata

5) AFFIDABILITÀ

È misurata attraverso due parametri:
MTTF: tempo medio di vita prima che si verifichi un guasto permanente
MTBF: frequenza media di guasti transitori
le memorie ad accesso casuale

- Caratteristiche generali:
  Ogni cella può essere indirizzata indipendentemente
  tempi di accesso uguali e costanti per ogni cella.
- Segnali di controllo permettono di riconoscere se:
  bisogna leggere o scrivere
  quando gli indirizzi sono disponibili sul BUS
  quando i dati sono disponibili sul BUS (per la scrittura o per la lett)

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 51 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 53 di 146

Difetto: La ROM non può più essere modificata! Pertanto qualsiasi errore di progettazione è drammatico.
Per evitare di perdere tutto il lavoro esistono delle piccole varianti che permettono di modificare il contenuto. Ad esempio le PROM non vengono programmate al momento della produzione ma al momento del loro utilizzo poiché al posto dell'architetto c'è un diodo che viene bruciato o meno a seconda se si vuole un circuito aperto o un corto circuito. Le EPROM sono ulteriormente riscrivibili, possono essere modificate un numero indefinito di volte e possono essere programmate anche ~~in fase di~~ se già montata nella scheda. Hanno un meccanismo di programmazione attraverso luce ultravioletta. Le EEPROM non hanno bisogno della luce ultravioletta. Le memorie FLASH sono come le EEPROM ma con tempi di scrittura un po' più veloci delle EEPROM e vengono scritte a blocchi.

RAM

~~Sono~~ Sono molto più veloci delle altre memorie, ma la loro caratteristica base è l'essere volatili. Esistono due tipologie di memorie RAM:

STATICHE: Per implementare una singola cella si usa un flip-flop
Queste sono molto veloci ma non possono memorizzare molti bit

DINAMICHE: Utilizzano un condensatore, più lento ma capace di memorizzare molte più informazioni.

Servono 6 transistor per memorizzare ogni bit di una memoria RAM statica. Nella memoria dinamica vengono utilizzati uno transistore in modo tale che a parità di area di silicio si possa avere una memoria più capiente.
Al posto dell'architetto c'è un condensatore, il meccanismo funziona tanto è più piccolo il condensatore che utilizzo in modo tale da poter utilizzare molti più condensatori. Inoltre abbiamo bisogno di una circuiteria che continuamente esegua il meccanismo di refresh altrimenti si perde il contenuto. Abbiamo ottenuto una memoria che per ogni bit utilizza un solo transistor. Ma perché è più lento? Perché le operazioni di lettura e scrittura devono caricare o scaricare dei condensatori.

Rinfresco: Consiste nell'operazione di amplificazione del contenuto del condensatore. Si basa su operazioni di lettura fittizie poiché il valore non viene trasmesso. Durante il rinfresco si possono eseguire le operazioni normalmente, ma se devo accedere alla cella in corso di rinfresco ~~deve~~ devo aspettare.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 53 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 55 di 146

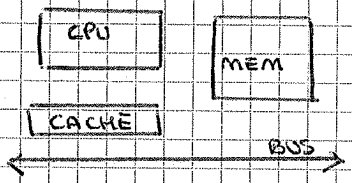
In termini di prestazioni questo approccio è notevolmente più favorevole:

$t_{medio} = hC + (1-h)M$

h indica la probabilità che nella cache ho le informazioni che mi servono
C è il tempo di accesso alla cache
M rappresenta la penalità di fallimento ovvero il tempo di accesso alla memoria + il caricamento del blocco in cache.
Principale. La struttura della cache deve avere un meccanismo per rilevare le informazioni maggiormente utilizzate. La cache contiene anche una parte chiamata tag che memorizza l'indirizzo del blocco che in quell'istante è contenuto nella cache. Pertanto tag è una tabella che mi dice per ogni linea quale è il blocco a cui appartiene. Tutto ciò è accompagnato da una parte logica che si interfaccia con il processore.

HIT: con il termine si indica quel fenomeno per cui l'informazione si trova in cache

MISS: si indica il caso in cui l'informazione non si trova già in cache. In questo caso bisogna accedere in memoria e oltre a questo caricare attraverso la cache contro il blocco. Pertanto il sistema in questo caso raggiunta ancor di più del caso di lettura semplice dalla memoria. In realtà la frequenza dei miss non è molto elevata. In alcuni casi quindi si preferisce non usare il meccanismo di cache.



La cache è messa in posizione strategica per evitare quando possibile di utilizzare il BUS.

La cache è divisa in cache istruzioni e cache dati in alcuni casi e in questo caso si usa un'architettura di tipo Harvard.

Parametri della cache:

- Dimensioni:
- Meccanismo di mapping: la cache deve decidere quali blocchi caricare in memoria. Quindi deve accedere alla mem. principale e caricarlo, dopo ciò deve mettere l'indirizzo del blocco nel tag.
  Il problema è dove metto il blocco nella cache già tutta piena?
  Meccanismo del direct mapping: all'inizio i blocchi vengono messi nell'indice corrispondente cioè blocco 0 in posizione 0, blocco 1 in posizione 1
  In termini matematici: $K = i \bmod N$ la cache esegue in tempo estremamente basso il calcolo di k e sistema il blocco.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 55 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 57 di 146

Set associative mapping

L'idea è quella di dividere la cache in insiemi di 2, 4, 8 linee. Nel momento in cui devo accedere K ho fatto nella stessa maniera K = i mod N* una stanghetta e K mi indica la linea nell'insieme, e poi con meccanismo pseudo-casuale metto nell'insieme K e nella riga i il codice.

* N = numero degli insiemi

I confronti sono pari al numero di linee facenti parte dell'insieme ad esempio 4 linee - 4 comparatori.

Caso di miss:

In caso di miss il processore va in memoria e prende il blocco, tuttavia per rispettare il principio di località esso deve prende il blocco meno utilizzato e sostituirlo (alcuni metodi utilizzano la strategia FIFO per vedere quale blocco è presente nella cache da più tempo, ma questo meccanismo è poco intelligente). Un meccanismo molto efficiente utilizza le LFU (quello poco acceduto viene cancellato) ma è troppo costoso! La strategia vincente è quella del LRU sostituisce quello acceduto meno recentemente cioè quale linea del blocco è stata acceduta per prima. La differenza con la FIFO è che questo guarda tutto il tempo dall'inizio non solo negli ultimi accessi.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 57 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 59 di 146

Intel 8255

Interfaccia Parallela Programmabile

Il processore deve poter comunicare con le periferiche I/O. Lo fa attraverso l'uso delle interfacce, un esempio potrebbe essere la porta parallela programmabile. Esso viene impiegato per la famiglia 8085 e 8086

CHIP: È costituito da un corpo e tanti pin (piedini) che possono andare verso il processore o verso l'esterno attraverso il bus.

- Indirizzo Porta

Insieme di registri da 8 bit corrispondenti ~~ai~~ alle 3 porte e al registro di controllo. Il metodo utilizzato per la gestione delle periferiche è del tipo isolated I/O.

Accedendo ai tre registri delle Porte si esegue il trasferimento dati da essi posso ricevere e scrivere. Il registro di controllo permette di gestire cosa voglio fare sulle altre porte. ~~Se~~ su di esso si può soltanto scrivere. L'indirizzo della Porta parallela è definito al momento della "costruzione" è quindi un qualcosa di cablato non è deciso dal programmatore.

- Segnali di controllo

CS: Abilita la comunicazione tra CPU e 8055 attraverso un segnale basso.

RD: Inviare il dato o l'informazione sul data bus per la CPU

Esistono tanti altri segnali di controllo. (WR, RESET)

- Modi di Funzionamento

- Modo 0
- Modo 1
- Modo 2

} lavorano con l'interrupt controller

I Pin sono raggruppati in due gruppi (A, B)

Le parole di controllo sono scritte sul quarto registro e possono avere due funzioni:

a) Programmazione del modo di funzionamento

b) Scrittura bit a bit (di un singolo valore logico in un singolo bit della Porta C).

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 59 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 61 di 146

MECCANISMI DI I/O

I/O Programmato

È gestito completamente a livello software, quando la parola deve essere scritta la CPU interroga il registro di stato, fin quando questo non indica che il dato è stato caricato. Il problema è che è poco efficiente perché la CPU deve "aspettare" che la periferica abbia terminato, senza far altro.

Interrupt

In questo caso la periferica comunica con la CPU tramite un bit che assume un determinato valore a seconda se la periferica è pronta a ricevere un nuovo dato. Quando questo bit è a uno, c'è un altro segnale, quello di interrupt che "ferma" la CPU (Procedura di servizio dell'interrupt)
In questo caso il processore può far altro mentre la periferica lavora. Se voglio gestire più periferiche (e il segnale di interrupt è unico) allora devo avere anche un interrupt controller ovvero un dispositivo che inoltra la richiesta di servizio alla CPU. Come fa la CPU a capire qual è la periferica che ha richiesto l'interrupt? Esistono un certo numero di procedure di servizio dell'interrupt a seconda della periferica pronta se ne attiva una e la CPU svolge i compiti giusti. Inoltre ogni periferica ha un codice proprio, quando la CPU si accorge che l'interrupt controller è a 1 attiva l'interrupt acknowledge e a sua volta l'interrupt controller carica il codice della periferica. Esiste una interrupt vector table in cui per ogni elemento c'è scritto l'indirizzo della ISR associata alla periferica. Quindi un reset dopo l'interrupt acknowledge, l'interrupt controller fornisce il numero della periferica, che è anche il numero della cella dell'interrupt vector table (situata di norma a partire dall'indirizzo 0)
Quando la CPU sta eseguendo un programma e arriva la richiesta da una periferica, quando finisce di eseguire la periferica deve ritornare al programma che stava eseguendo, lo fa attraverso un meccanismo simile a

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 61 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 63 di 146

Inconveniente: mentre il DMA lavora, la CPU perde l'utilizzo del bus, ed è quindi costretta ad aspettare il DMA.
Per risolvere le problemi si utilizzano tre modi:

- trasferimento a blocchi
- trasferimento con Cycle Stealing
- trasferimento in Trasparent DMA

I BUS

Il bus è una struttura di comunicazione generalmente condivisa. L'interfaccia è composta da due moduli insieme dati da un'unica struttura, le transceiver.

Bus unidirezionale

La linea di bus permette il passaggio sia di dati che di indirizzi, questo perché essi non vengono mandati contemporaneamente, bisogna solo far attenzione a capire che tipo di dati stanno viaggiando.

Bus multiplo

Ci sono due linee di bus differenti, in modo tale da poter mandare contemporaneamente dati o indirizzi a più dispositivi, CPU o memoria ecc...

Master e slave

Le unità connesse al bus sono di due tipi:

Master: il dispositivo decide quando iniziare, come accedere ecc... cioè prende il controllo del bus

Slave: risponde ai comandi del dispositivo master

Quando ci sono più master bisogna stabilire chi sia quello effettivo.

Sincroni, Asincroni

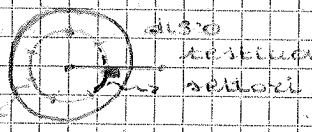
I bus sincroni sono quelli che condividono lo stesso segnale di clock (che fa parte del bus stesso) la frequenza del clock è determinata dal dispositivo più lento, ogni dato deve rassomigliare la destinazione nello stesso tempo delle altre, pertanto un bus di questo tipo non può essere troppo lungo.
Inconveniente: se ho moduli che reagiscono con tempi diversi devo adattare la frequenza a quello più lento, rallentando anche gli altri.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 63 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 65 di 146

Polling: All'arbitro arriva l'or dei segnali dei dispositivi quindi un solo bus request.
L'arbitro manda sue "bus" (linee di sotto) le bus grant. (poll counter in numero log2n con n= numi dispositivi)
L'arbitro interroga le unità per sapere chi ha fatto richiesta, con un ordine casuale
La differenza con il daisy chaining è che nel primo si trasmette le segnale uno secondo è l'arbitro che interroga.

Memorie ad accesso seriale
Sono collocate nella parte più bassa della gerarchia delle memorie. La loro caratteristica fondamentale è l'elevato tempo di accesso



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 65 di 146

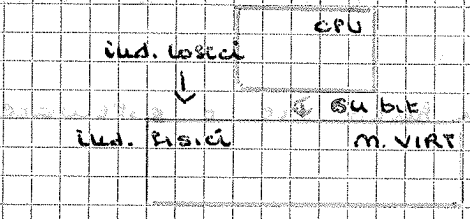
© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 67 di 146

La memoria virtuale

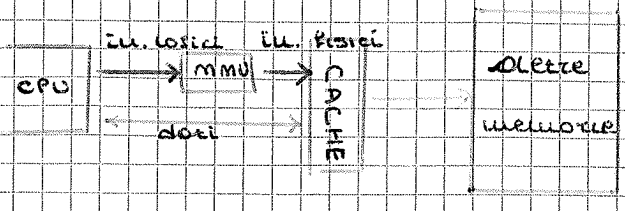
Per quanto riguarda le memorie esistono diversi livelli classificabili attraverso due parametri principali:

- dimensione
- costo

Ulteriori parametri sono ad esempio il tempo di accesso alla memoria. Scendendo nei livelli aumenta il tempo d'accesso pertanto tra mem. principale e secondaria è opportuno inserire un livello costituito dalla memoria virtuale simile alle cache (tranne per il fatto che la cache si interfaccia tra registri e mem. principale).
Il meccanismo della memoria virtuale è alquanto diverso da quello delle cache.



Il meccanismo sussiste nel far credere alla CPU che esiste un'unica grande memoria virtuale che mi crede una memoria principale anche se così non è. Abbiamo bisogno di un dispositivo che traduce gli indirizzi emessi che la CPU invia (perché crede che la memoria sia emessi) in indirizzi fisici con cui realmente si può accedere alla mem. principale, se poi questi non sono in memoria, deve fermare la CPU. Tale modulino prende il nome di memory management unit. Nel caso in cui non sia presente l'indirizzo in memoria princ. si parla di page fault (equivalente di miss). L'MMU si accorge che la pagina manca e chiama il sistema operativo che prende questi indirizzi mancanti. La dimensione dei blocchi rispetto alle cache è notevolmente maggiore.



L'MMU quando si accorge di page fault chiama il meccanismo di eccezione.

Nell'indirizzo logico i bit alti rappresentano la pagina di mem. virtuale quelli bassi la riga della pagina.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 67 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 69 di 146

Le architetture a pipeline

Sono architetture di processori più avanzati.

Per aumentare le prestazioni di un processore si può:

a. aumentando la velocità dei componenti (fino ad un certo punto sfruttavano solo le tecnologie, es. i transistor i chip...)

b. cambiando l'architettura del processore

Il numero di colpi di clock per eseguire un'istruzione dipende dall'architettura, il tempo di ogni istruzione dipende dalla freq. a cui lavora il processore.

Pipeline

- È l'equivalente della catena di montaggio. Ogni istruzione è articolata in diverse fasi: FATCH, DECODE, EXECUTE, WRITE. Il processore esegue le fasi di un'istruzione, poi quelle della seguente ecc. e nel processore una parte si occupa del fatch uno dell'execute ecc., pertanto lavora una sola parte per volta. Si è pensato quindi di far funzionare contemporaneamente tutte le parti del processore. Ad ogni colpo di clock tutte e 4 le unità lavorano su qualcosa. L'hardware che si occupa di una singola istruzione è chiamato stadio. Dopo un certo numero di clock ad ogni colpo di clock si completa un'istruzione.

- Presupposto: (1 istr. ogni clock, stesso set di istruzioni)
  Il set di istruzioni da eseguire deve essere lo stesso, o perlomeno simile, non posso mettere insieme una NOP con una DIV! La NOP terminerà molto prima! Quindi ogni stadio deve essere eseguito nello stesso tempo in cui vengono eseguiti gli altri. Esistono dei registri di pipeline in cui vengono memorizzati i risultati dello stadio precedente.
  CISC il numero di istruzioni completate ad ogni colpo è <1
  RISC completano un'istruzione ogni colpo di clock
  SUPERSCALARI riescono a completare anche più istruzioni al colpo di clock

- Problema: I presupposti sono difficili da verificare!
  Il problema più grande delle pipeline è lo stallo cioè il processore non riesce a completare l'istruzione in un solo colpo di clock.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 69 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 71 di 146

Questo si verifica perché due istruzioni contengono un dato in dipendenza, cioè la seconda contiene un dato che dipende dalla prima.

Soluzioni:

1) Rendere più complesso il processore in modo che capisca se c'è una dipendenza e introduca degli stalli (Soluzione hardware)

2) L'hardware non viene modificato, ma chi scrive il codice sa come è fatto un processore e assume delle NoP per fardare l'istruzione successiva. In realtà non è il programmatore ad assumere le NoP ma il compilatore. Se cambio il processore il codice non va bene! (Soluzione software)

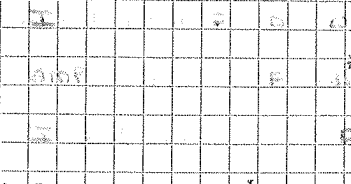
- Problema 3: Istruzioni di salto!

Quando si incontra un'istruzione di salto, l'aggiornamento del program counter avviene nella fase di write, tuttavia anche se si dovrebbe "saltare" ad eseguire lab, il processore ha già avviato un'altra istruzione che non sarà quella di label poiché il program counter non è aggiornato! Quindi nella pipeline entrano un numero n di istruzioni scorrette.

Soluzioni:

1) Viene modificato l'hardware in modo che non appena il processore carica un'istruzione di salto, blocca il fetch e le fasi successive delle altre istruzioni

Vediamo come:



* Il processore si accorge del salto solo dopo aver ~~eseguito~~ eseguito il fetch dell'istruzione seguente, pertanto dovrà essere eliminata un'istruzione dalla pipeline (Soluzione hardware)

2) Dopo l'istruzione di salto devo assumere un numero n (branch delay slot) di NoP (Soluzione software)

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 71 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 73 di 146

3 Processori superscalari

Possono completare più di un'istruzioni per ogni colpo di clock. Sono costituiti da più pipeline.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 73 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 75 di 146

es. Consideriamo 10 partecipanti ad una gara, voglio trovare il numero di possibili podi cioè di risultati per i 1° 2° 3° posto

$D_{10,3} = \frac{m!}{(m-k)!} = \frac{10!}{7!} = 10 \cdot 9 \cdot 8$ con $m=10$ $k=3$

Caso particolare: Se voglio calcolare $D_{m,m}$ cioè se $k=m$ allora si parla di permutazioni e si indica con $\boxed{P_m = m!}$

Ipotesi: Fin'ora abbiamo considerato che nell'estrazione in cui un oggetto veniva scelto, questo poi veniva messo da parte, se invece non lo eliminiamo dall'insieme delle possibili scelte, otteniamo le cosiddette disposizioni con ripetizione e lo indicheremo con

$$\boxed{D^{(r)}_{m,k} = \underbrace{m \cdot m \cdot \ldots \cdot m}_{k \text{ volte}} = m^k}$$

esempio: Numeri binari: ho k bit quanti numeri posso formare?

n° rappresentazioni $= 2^k$

- Permutazioni con elementi indistinguibili

es. Consideriamo la parola ALA quanti anagrammi?

AAL
ALA
LAA
} 3 anagrammi

Ma in realtà quello che sto facendo non è altro che una permutazione e noi sappiamo che $P_m = m!$ in questo caso otterrei $P_3 = 3! = 6$ come mai invece gli anagrammi sono solo 3?

- Se indico le lettere A con A_1 e A_2 ottengo:

A_1A_2L
A_1LA_2
A_2LA_1
A_2A_1L
LA_1A_2
LA_2A_1
} 6 anagrammi

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 75 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 77 di 146

(es.) Abbiamo un mazzo di 20 persone ne prendo 2.

Quante sono le possibili coppie

$$\#(20,2) = \binom{20}{2} = \frac{20!}{2!\cdot 18!} = \frac{20\cdot 19}{2} = 190$$

Esercizi

1. Consideriamo tutte le possibili coppie (i,j) $i,j \in \{0,1,\dots,9\}$

a) Cardinalità dell'insieme?

b) Quante di esse sono con $i \neq j$?

c) Quante con $i < j$?

d) Quante con $i \leq j$?

Svolgimento

a. $n = 10^2 = 10\cdot 10$ è una disposizione con ripetizione

b. $n = 10\cdot 9 = 90 = D_{10,2}$ (disposizione)

c. $n = \frac{90}{2} = 45 = C_{10,2}$

d. $n = 45 + 10 = 55$

2. Trovare sei anagrammi della parola PALLA

$M_A = 2$ $M_L = 2$ $M_P = 1$ $M_{tot} = 5$

$$n = \binom{5}{2\,2\,1} = \frac{5!}{2!\,2!\,1!} = 30$$

3. Consideriamo un'urna con 20 palline numerate da 1 a 20 ed eseguiamo 5 estrazioni con reimbussolamento (rimetto dentro l'urna la pallina estratta). Quante sono le possibili sequenze?

$n = \underbrace{20\cdot 20\cdot \dots \cdot 20}_{5 \text{ volte}} = 20^5$

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 77 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 79 di 146

Probabilità (def)

Indichiamo con il termine esperimento, un evento di cui non conosco l'esito e denotiamo con Ω l'insieme dei possibili risultati. Denotiamo con $A \subseteq \Omega$ un possibile sottoinsieme es. risultato pari o dispari, comunemente si associa ad A la probabilità che esca quel risultato e si indica con $P(A)$ ed è compresa tra 0 e 1.

Concezione classica: $P(A) = \frac{\text{casi favorevoli ad A}}{\text{casi possibili}} = \frac{\text{cardinalità di A}}{\text{cardinalità di } \Omega}$

con tale definizione però stiamo supponendo che i casi abbiano tutti la stessa probabilità di verificarsi, cioè casi equiprobabili

Concezione frequentistica: $P(A)$ = limite supposto esistente, del rapporto tra casi favorevoli ad A e totale di prove effettuate

$$P(A) = \lim_{n \to \infty} \frac{n_A}{n}$$

Ma insorgono anche qui dei problemi, innanzitutto per la definizione di limite dovremmo avere un intorno di n, ma non ce lo abbiamo, inoltre l'esperimento andrebbe ripetuto infinite volte. Le condizioni devono essere le stesse e l'esperimento si deve poter ripetere! (non sempre è possibile)

Concezione soggettivistica:

$P(A)$ = Misura del grado di fiducia che un individuo coerente attribuisce al verificarsi di A

Grado di fiducia: alta che sono disposto a scommettere per guadagnare 1 se A si verifica

Individuo coerente: disposto ad essere sia scommettitore che bookmaker.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 79 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 81 di 146

(es.) Sia $\Omega = \mathbb{R}$

$\mathcal{A} = \{ [a;b]$ con $a,b \in \mathbb{R}$ e $a<b$ più tutte le possibili unioni e intersezioni di questi intervalli $\}$

Consideriamo anche dei insiemi:

$\mathcal{A}_m = (a, b-1/n)$ con $n \in \mathbb{N}^+$

$\bigcup_{n=1}^{m} \mathcal{A}_n = (a;c]$ è sicuramente un'algebra ma non è una σ-algebra perché $\bigcup_{n=1}^{\infty} \mathcal{A}_n = (a;b)$ (non soddisfa la proprietà III delle σ-algebre)

DEF: Una applicazione $P: \mathcal{A} \to \mathbb{R}$ ovvero $P: A \in \mathcal{A} \to P(A) \in \mathbb{R}$ è detta <u>probabilità</u> se soddisfa 3 condizioni:

1. $P(A) \in \mathbb{R}^+$ cioè $P(A) \geq 0$
2. $P(\Omega) = 1$
3. Data famiglia $\{A_i\}_{i \in \mathbb{N}}$ di eventi incompatibili, se $\bigcup_{i=1}^{\infty} A_i \in \mathcal{A}$ allora $P\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} P(A_i)$

<u>Osservazione</u>: Alla proprietà 3 potremmo aggiungere una 3' così definita:

$A, B \in \mathcal{A}$, $A \cap B = \emptyset$ allora:

$P(A \cup B) = P(A) + P(B)$ (caso particolare della 3)

(Proprietà di addittività)

È detta <u>terna di probabilità</u> o spazio di probabilità la seguente

$(\Omega, \mathcal{A}, P)$

Le funzioni che ad un sottoinsieme associano un numero reale si dicono <u>misure</u>. Allora le probabilità sono particolari misure.

Esempi:

1) Supponiamo di avere un'urna con tre biglie, due di colore rosso e una bianca

<u>urna</u>: $\{R_1, R_2, B\}$

L'esperimento consiste nell'estrazione di una biglia

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 81 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 83 di 146

Esempio: Lancio dato equilibrato insieme delle facce di Ω

$\Omega = \{1, 2, \dots, 6\}$ $\mathcal{A} = \mathcal{P}(\Omega)$

P t.c $P(\{a\}) = 1/6$ è una probabilità uniforme in quanto ad ogni faccia del dado assegno usuale probabilità di verificarsi

- $A = \{$esce un numero pari$\}$ $\Rightarrow$

$P(A) = P(\{2\} \cup \{4\} \cup \{6\}) = P(\{2\}) + P(\{4\}) + P(\{6\}) = 3 \cdot \frac{1}{6} = \frac{1}{2}$

- $B = \{$esce un multiplo di 3$\}$ $\Rightarrow$

$P(B) = P(\{3\} \cup \{6\}) = P(\{3\}) + P(\{6\}) = \frac{1}{3}$

- $C = \{$esce un pari o un multiplo di 3$\}$ $\Rightarrow$

$P(C) = P(A \cup B) = P(A) + P(B) - P(A \cap B) = \frac{2}{6} + \frac{3}{6} - P(\{6\}) =$

$\frac{2}{6} + \frac{3}{6} - \frac{1}{6} = \frac{4}{6} = \frac{2}{3}$

Oppure:

$P(C) = P(\{2\}, \{3\}, \{4\}, \{6\}) = 4 \cdot \frac{1}{6} = \frac{2}{3}$

- $D = \{$esce un numero dispari$\}$

$P(D) = P(\{1\}, \{3\}, \{5\}) = 3 \cdot \frac{1}{6} = \frac{1}{2} = P(\Omega) - P(A) = 1 - \frac{1}{2} = \frac{1}{2}$

Esempio: 5 componenti indistinguibili di cui 3 funzionanti 2 guasti

estraiamo 2 di questi componenti.

- $A = \{$entrambi sono guasti$\}$
- $B = \{$estraggo solo 1 guasto$\}$
- $C = \{$almeno 1 è guasto$\}$

$\Omega = \{(g, g), (g, f), (f, g), (f, f)\}$

Oppure posso chiamare i dispositivi: g_1, g_2, f_1, f_2, f_3

$\Omega = \{(g_1, g_2), (g_1, f_1), \dots, (g_2, g_1), (f_1, g_1) \dots\}$

$\#\Omega = 5 \cdot 4 = 20$ $P(\{a, b\}) = \frac{1}{20}$

a. $A = \{(g_1, g_2), (g_2, g_1)\}$

$P(A) = P\{(g_1, g_2)\} + P\{(g_2, g_1)\} = 2 \cdot \frac{1}{20} = \frac{1}{10}$

$P(A) = \frac{\mu_A}{\mu} = \frac{2}{5 \cdot 4} = \frac{1}{10}$

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 83 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 85 di 146

• Probabilità Condizionata (def 16)

Siano $A, B \in \Omega$ e $P(B) \neq 0$

La probabilità condizionata dell'evento A, dato B, è la seguente:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Volendo fare un parallelo con la definizione classica di probabilità otteniamo che:

$P(A) = \frac{m_A}{m}$ uguale $P(A|B) = \frac{m_{A \cap B}}{m_B}$

Dimostrazione: $P(A|B) = \frac{m_{A \cap B}}{m_B}$

divido tutto per m: $P(A|B) = \frac{\frac{m_{A \cap B}}{m}}{\frac{m_B}{m}} = \frac{P(A \cap B)}{P(B)}$

Esempio: Lancio del dado (equilibrato) $\Omega = \{1, 2, \ldots 6\}$

$A = \{\text{esce } 1 \text{ o } 2\}$ $P(A) = 1/6$

$B = \{\text{esce pari}\}$

$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{1/6}{3/6} = \frac{1}{3}$

$\Omega_B = \{2, 4, 6\}$ $P(A|B) = 1/3$

• Relazioni tra $P(A)$ e $P(A|B)$

es: A = esce il numero 2
B = esce un num. pari

$P(A) = 1/6 \leq P(A|B) = 1/3$ (aumenta)

es: A = esce il numero 2
B = esce un dispari

$P(A) = 1/6 \geq P(A|B) = \frac{P(\phi)}{P(B)} = 0$ (diminuisce)

es: A = esce un numero $\in 2$
B = esce un dispari

$P(A) = 2/6 = P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{1/6}{3/6} = \frac{1}{3}$ (costante)

Definizione: Due eventi A e $B \in \Omega$ sono detti STOCASTICAMENTE INDIPENDENTI se $P(A|B) = P(A)$

Più precisamente se $P(A \cap B) = P(A) \cdot P(B)$

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 85 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 87 di 146

Teoremi e Proprietà (cc. 11)

PROP: Siano A e B indipendenti ⟹ $\bar{A}$ e $\bar{B}$ sono indipendenti

Dim: Sappiamo che: $P(A \cap B) = P(A) \cdot P(B)$

ma $P(\bar{A} \cap \bar{B}) = P(\overline{A \cup B})$ per De Morgan

$P(\bar{A} \cap \bar{B}) = 1 - P(A \cup B) = 1 - P(A) - P(B) + P(A \cap B) =$

$1 - P(A) - P(B) + P(A) \cdot P(B) = 1 - P(A) - P(B)(1 - P(A)) =$

$(1 - P(A))(1 - P(B)) = P(\bar{A}) \cdot P(\bar{B})$

Allora $\bar{A}$ e $\bar{B}$ sono indipendenti

Analogamente se A e B sono indipendenti, lo saranno

pure $A, \bar{B}$ oppure $\bar{A}$ e B

Teorema del Prodotto

Data una Partizione di eventi $\{A_1 \dots A_n$ con $A_i \neq \emptyset\}$ vale:

$P(A_1 \cap A_2 \cap \dots A_n) = P(A_1) \cdot P(A_2 | A_1) \cdot P(A_3 | A_1 \cap A_2) \dots \cdot P(A_n | A_1 \cap \dots A_{n-1})$

Dim (per induzione)

con $n=2$ $P(A_1 \cap A_2) = P(A_1) \cdot P(A_2 | A_1)$

$P(A_2 | A_1) = \frac{P(A_1 \cap A_2)}{P(A_1)}$ moltiplico ambo i membri

per $P(A_1)$ e ottengo il risultato

con $n > 2$ Supponiamo sia valido per $n-1$

$P(A_2 \cap \dots A_{n-1}) = P(A_2) \cdot P(A_3 \cap A_2) \cdot \dots \cdot P(A_{n-1} | A_2 \dots A_{n-2})$

Consideriamo $P(A_1 \cap \dots A_n) = P(B \cap A_n)$ dove $B = A_1 \cap A_2 \dots A_{n-1}$

Allora ci siamo ricondotti al caso $n=2$:

$P(A_1 \cap \dots \cap A_n) = P(B) \cdot P(A_n | B) =$

$P(A_1) \cdot P(A_2 | A_1) \cdot \dots \cdot P(A_{n-1} | A_1 \dots A_{n-2}) \cdot$

$P(A_n | A_1 \cap \dots A_{n-1})$

q.e.d

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 87 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 89 di 146

esempio: Immaginiamo di possedere 2 monete indistinguibili tali che solo la prima è equilibrata.

2 monete
- M1
  - Testa: 1/2
  - Croce: 1/2
- M2
  - Testa: 1/3
  - Croce: 2/3

Estraiamo una moneta ed effettuiamo un lancio

$\Omega = \{$ testa, croce$\}$ oppure $\Omega = \{(M1,T),(M2,T)(M1,C),(M2,C)\}$

$B = \{$esce testa$\} = \{(M1,T),(M2,T)\}$

- Consideriamo la partizione:
  A_1: esce M1
  A_2: esce M2

$P(B) = P(A_1) \cdot P(B|A_1) \oplus P(A_2) \cdot P(B|A_2) = \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{3} = \frac{5}{12}$ (oppure)

esempio: 2 dadi equilibrati e lancio in sequenza e calcolo la somma

$P(\text{somma} = 5) = ?$

$\Omega = \{2,3,4,\ldots,12\}$ Ma in questo caso i risultati non avrebbero uguale probabilità quindi non mi conviene utilizzare tale scelta per Ω.

$\Omega = \{(1,1),(1,2)\ldots(2,1)(2,2)\ldots\}$ $\quad \#(\Omega) = 36$

$P(B) = P(\{(a;b): a+b=5\}) = P(\{(1,4),(2,3),(3,2),(4,1)\}) = 4/36$

Ma questo approccio per tanti dadi non è comodo

A_1 = 1° dado = 1
A_2 = 1° dado = 2
!
A_6 = 1° dado = 6
} Partizione

$P(B) = \sum_{i=1}^{6} P(A_i) \cdot P(B|A_i) = P(A_1) \cdot P(B|A_1) + \ldots + P(A_6) \cdot P(B|A_6)$

$\frac{1}{6} \cdot \frac{1}{6} + \frac{1}{6} \cdot \frac{1}{6} + \frac{1}{6} \cdot \frac{1}{6} + \frac{1}{6} \cdot 0 + \frac{1}{6} \cdot 0 = \frac{4}{36}$

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 89 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 91 di 146

$P(B|D_2) = P(B_1 \cap B_2 | D_2) = P(B_1|D_2) \cdot P(B_2|B_1 \cap D_2)$

(per la formula del prodotto)

$= \frac{2}{3} \cdot \frac{1}{4}$

- $P(C) = P(\overline{B_1} \cap \overline{B_2}) = 1 - P(\overline{B_1} \cap \overline{B_2}) =$

$1 - [P(D_1) \cdot P(\overline{B_1} \cap \overline{B_2} | D_1) + P(D_2) \cdot P(B_1 \cap B_2 | D_2)]$

$1 - \left[\frac{1}{2} \cdot \frac{3}{2} \frac{2}{4} + \frac{1}{2} \cdot \frac{3}{6} \cdot \frac{2}{5}\right] = 1 -$

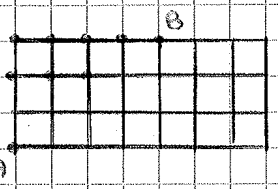
- $P(D_1 | 2 \text{ rosse}) = P(D_1|B) = \frac{P(D_1) \cdot P(B|D_1)}{P(B)} = \frac{1/2 \cdot 2/5 \cdot 1/4}{3/20}$

- $P(2^\circ \text{ rossa} | 1^\circ \text{ rossa}) = P(B_2|B_1) = \frac{P(B_1 \cap B_2)}{P(B_1)} =$

$= \frac{P(B)}{P(D_1) \cdot P(B_1|D_1) + P(D_2) \cdot P(B_1|D_2)} = \frac{3/20}{1/2 \cdot 2/5 + 1/2 \cdot 2/12} = \frac{1}{3}$

- Complementi Combinatorica (1632 u)

esempio:



Abbiamo una scala vogliamo calcolare il numero di percorsi possibili per andare da A a B spostandoci solo verso destra e in alto.

Quanti sono i percorsi possibili?

- Sicuramente ce ne dovranno essere 3 verso l'alto e 4 verso destra, tutti i percorsi sono sei anagrammi con elementi indistinguibili

U U U D D D $\Rightarrow$ $\#(P) = \frac{7!}{3! \cdot 4!} =$

Coefficienti multinomiali

Abbiamo n individui da suddividere in r gruppi, $n_1, n_2 \dots n_r$ con n_i = n° individui $n_1 + n_2 + \dots + n_r = n$

#(possibili suddivisioni) = ?

1° gruppo $\binom{n}{n_1}$ ... r° gruppo $\binom{n_r}{n_r} = 1$

2° gruppo $\binom{n-n_1}{n_2}$

⋮

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 91 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 93 di 146

(ii) $n=3$ $P(\text{non st. stesso}) = 1 - \frac{364}{365}\frac{363}{365} = 1 - \frac{364\cdot 363}{365^{n-1}}$

$= 1 - \frac{365\cdot 364\cdot 363\cdot 362!}{365^n (365-n)!}$

$= 1 - \frac{365!}{365^n (365-n)!}$

(iii) $n=23$ $p = 0{,}367$

esempio: Overbooking

ipotesi: • indipendenza tra i passeggeri

• $p \in (0,1)$ probabilità che il singolo passeggero non si presenti

• Preferair (compagnia piccola) AO → CN con 20 posti
Prenotazioni → 22 posti

$P(\text{overbook}) = ?$

Sia N il numero di individui presenti alla partenza.

$P(\text{overb}) = P[N>21] = P(\{N=21\} \cup \{N=22\})$

$= P(\{N=21\}) + P(\{N=22\}) = (1-p)^{22} + 22\,p(1-p)^{21}$

$= \binom{22}{22}(1-p)^{22}p^0 + \binom{22}{21}(1-p)^{21}p^1 + \dots$

• Variabili casuali (Le 25)

Continuità della misura di probabilità

Sia $P: \mathcal{A} \to \mathbb{R}$ cioè $A \longmapsto P(A) \in \mathbb{R}^+$

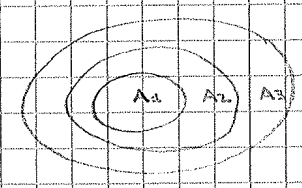
Proprietà: $\lim_{m\to\infty} P(A_m) = P\left(\lim_{m\to\infty} A_m\right)$

Definizione: Siano $\{A_m,\ m \in \mathbb{N}^+\}$ $A_m \in \mathcal{A}$

Sia A_m una successione monotona crescente

$A_m \subseteq A_{m+1}$ $\forall m \in \mathbb{N}$

Allora $\lim_{m\to\infty} A_m = \bigcup_{m=1}^{\infty} A_m$



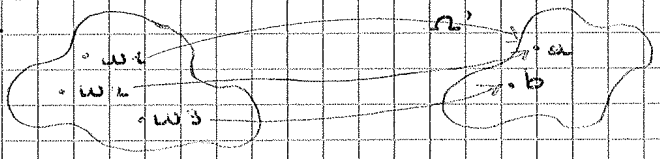
Non vuole che ogni avanti con n ottenga un insieme più grande allora il limite sarà allora più grande possibile.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 93 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 95 di 146

esempio: $(\Omega, \mathcal{A})$ $(\Omega', \mathcal{A}')$

Ω Ω'



$\mathcal{A} = \{\emptyset, \Omega, \{\omega_1, \omega_2\}, \{\omega_3\}\}$ $\mathcal{A}' = \{\emptyset, \Omega, \{a\}, \{b\}\}$

R: $\omega_1 \longmapsto a$
$\omega_2 \longmapsto a$
$\omega_3 \longmapsto b$

È misurabile? Verifichiamo:

$R^{-1}(\{a\}) = \{\omega_1, \omega_2\} \in \mathcal{A}$
$R^{-1}(\emptyset) = \emptyset \in \mathcal{A}$
$R^{-1}(\{b\}) = \{\omega_3\} \in \mathcal{A}$

} Sì è misurabile!

esempio: $(\Omega, \mathcal{A})$ $(\Omega', \mathcal{A}')$

Ω Ω'

$\mathcal{A} = \{\emptyset, \Omega, \{\omega_1, \omega_2\}, \{\omega_3\}\}$ $\mathcal{A}' = \{\emptyset, \Omega, \{a\}, \{b\}\}$

R: $\omega_1 \longrightarrow a$
$\omega_2 \longrightarrow b$
$\omega_3 \longrightarrow b$

È misurabile?

$R^{-1}(\{b\}) = \{\omega_2, \omega_3\}$ non contenuto in $\mathcal{A}$

$\Rightarrow$ R non è misurabile!

Definizione: Data spazio $(\Omega, \mathcal{A}, P)$ è detta variabile casuale una funzione:

$X: \Omega \to \mathbb{R}$ $X: \omega \longrightarrow X(\omega) \in \mathbb{R}$

tale che sia misurabile rispetto all'algebra $\mathcal{B}$ definita su $\mathbb{R}$, cioè t.c. $\forall B \in \mathcal{B}$, $X^{-1}(B) \in \mathcal{A}$

$(\Omega, \mathcal{A}, P) \xrightarrow{X} (\mathbb{R}, \mathcal{B}, P_X)$

P_X è definita come: $\forall B \in \mathcal{B}$ $P_X(B) = P[\{\omega \in \Omega : X(\omega) \in B\}]$

$= P[X^{-1}(B)]$

(definito perché X è misurabile)

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 95 di 146

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 97 di 146

Variabili aleatorie ——— discrete
assolutamente continue
continue
miste

• Discrete (v.c. 2.6)

Una v.c è detta discreta se e l'insieme S dei valori da essa potenzialmente assumibili è di cardinalità finita o numerabile.

S = supporto della variab. cas.

es: X = risultato del lancio di un dado

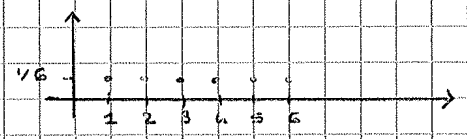
$S = \{1, 2, 3, \dots 6\}$

Densità discreta di probabilità:

$$f_X(t) = \begin{cases} P[X=t] & \text{se } t \in S \\ 0 & \text{altrimenti} \end{cases}$$

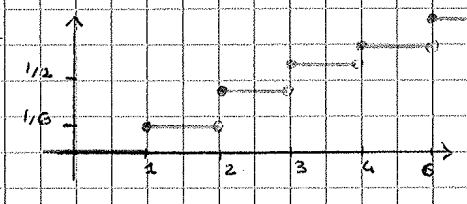
es: dado equilibrato

$$f_X(t) = \begin{cases} 1/6 & \text{se } t \in \{1, 2 \dots 6\} \\ 0 & \text{altrimenti} \end{cases}$$



$$F_X(t) = P[X \le t] = \sum_{t_i \in S,\ t_i \le t} f_X(t_i)$$ (funzione di ripartizione)

Funzione a gradino



$t < 1 \quad F_X(t) = 0$

$t \in [1,2) \quad F_X(t) = 1/6$

$t \in [2,3) \quad F_X(t) = 2/6$

$t \in [5,6) \quad F_X(t) = 5/6$

$t \ge 6 \quad F_X(t) = 1$

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 97 di 146