Sorgente : http://holsi.hasanaj.com/
Autore : Holsi Hasanaj
Professore: Riccardo Torlone
Corso di Ingegneria Informatica Roma Tre
Anno di produzione: 2011-2012

Calcolatori Elettronici

-Riassunti Libro-

Calcolatori Elettronici

ARCHITETTURA DEI CALCOLATORI
CAP 1 1/

LINGUAGGIO MACCHINA: insieme di istruzioni primitive → SOMMARE due numeri → controllare se un numero vale 0 → copiare una porzione di dati da una parte all'altra della memoria

↳ si utilizza un approccio strutturale poiché il ling. macchina è troppo difficile e tedioso

Si tende quindi a suddividere in varie gerarchie di linguaggi più comprensibili si fa uso di
- TRADUZIONE
- INTERPRETAZIONE → il programma che la esegue è detto interprete
- MACCHINA VIRTUALE: riassume in un unico concetto traduzione ed interpretazione

La definizione di una serie di linguaggi, ciascuno dei quali più pratico da utilizzare rispetto al precedente può continuare indefinitamente finché non se ne ottiene uno sufficientemente adeguato

→ Una macchina ha un proprio linguaggio macchina, costituito da tutte le istruzioni che è in grado di eseguire di fatto una macchina def. un linguaggio

LIVELLO 0 - **LIVELLO LOGICO DIGITALE**: sono le porte che utilizzano componenti logici come transistors
- combinando opportunamente le porte posso formare → memorie che combinate mi danno i registri

LIVELLO 1 - **LIVELLO DI MICROARCHITETTURA**: è presente una memoria locale formata da un gruppo di registri e un circuito chiamato ALU per effettuare ~~me~~ semplici operazioni aritmetiche
Inoltre i registri sono connessi all'~~de~~ ALU per formare un percorso dati lungo le quale quest'ultimi si spostano

LIVELLO 2 - **LIVELLO DI ARCHITETTURA DELL'INSIEME DI ISTRUZIONI**

ISA: Istruction Set Architecture

LIVELLO 3 - **LIVELLO DI ARCHITETTURA DELL'INSIEME DI ISTRUZIONI**: le istruzioni nel livello 3 sono realizzati da un interprete eseguito al livello 2 → chiamato SISTEMA OPERATIVO

LIVELLO 4 - **LIVELLO DEL LINGUAGGIO ASSEMBLATIVO** tra questo livello ed il precedente → spaccatura in quanto da qui in poi vengono orientate verso la programmazione applicativa
- Linguaggio ASSEMBLATIVO è una forma simbolica di uno dei sottostanti
- Il linguaggio assemblativo vengono quindi effettuati ai livelli inferiori → qui avviene la traduzione del programma che prende il nome ASSEMBLATORE
- i programmi scritti in a linguaggi di alto livello → vengono tradotti ai livelli più bassi tramite LA COMPILAZIONE

LIVELLO 5 - **LIVELLO DEL LINGUAGGIO ORIENTATO AL TIPO DI PROBLEMA** consiste di un interprete specifico per un determinato campo di applicazione

→ ARCHITETTURA: è l'insieme dei tipi di dati, delle operazioni e della funzionalità di ciascun livello

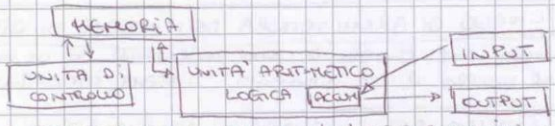
↳ HARDWARE: consiste degli oggetti tangibili
↳ SOFTWARE: consiste di algoritmi e delle loro applicazioni

(!) è hardware e software sono logicamente equivalenti: "L'hardware non è che software pietrificato"

GENERAZIONE 0: (1623-1662) Il primo che costruì una macchina calcolatrice funzionante fu Pascal → macchina interamente meccanica costituita da ingranaggi e azionata a mano e sapeva eseguire solo addizioni e sottrazioni. Poi Leibniz ne implementò facendole compiere anche le moltiplicazioni e le divisioni.
Nei primi 1800 Babbage costruì "la difference engine" che sapeva eseguire un solo algoritmo → poi costruì "l'analytical engine": interamente meccanica, consisteva nel fatto di essere di uso generale ovvero leggeva le istruzioni dalle schede perforate e le eseguiva.
Fu quindi la prima macchina programmabile attraverso un linguaggio assemblativo
- Babbage fu il padre degli odierni calcolatori

Negli anni 30' Zuse costruì una serie di macchine calcolatrici automatiche utilizzando dei relè elettromagnetici → scoppio guerra → no realizzazione
Macchina di Atanasoff → basata sull'aritmetica binaria e utilizzava dei condensatori per la memoria

GENERAZIONE 1: La prima generazione di calcolatori ebbe un largo sviluppo durante la seconda guerra mondiale. - Colossus fu il primo elaboratore digitale al mondo in grado di decifrare il codice enigma. Mauchley costruì un calcolatore elettronico → "ENIAC" (Electronic Numerical Integrator And Computer) era costituito da 18000 valvole termoioniche e 1500 relè pesava 30 tonnellate e consumava 140 kW di energia. Insegnato Mauchley realizzò una versione perfezionata della precedente macchina "EDVAC" (Electronic Discrete Variable Automatic Computer).
Von Neumann un genio di livello di Leonardo da Vinci, parlava molte lingue, era un esperto di Fisica e Matematica. Una cosa che gli apparve subito evidente fu che la programmazione dei computer mediante un gran numero di interruttori e cavi era lenta, tediosa e poco flessibile
↳ comprese che i programmi potevano essere rappresentati in forma numerica all'interno della memoria del computer
↳ Il progetto di VON NEUMANN fu utilizzato nel computer EDSAC, il primo che memorizzava le programma in memoria
- LA MACCHINA di VON NEUMANN = era composta di 5 componenti principali:

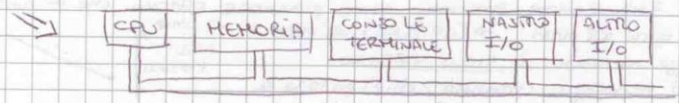


- LA MEMORIA era costituita da 4096 locazioni ognuna conteneva 40 bit
- UNITÀ ARITMETICO-LOGICA si svolgevano le operazioni aritmetiche, inoltre all'interno era presente un ACCUMULATORE → speciale registro da 40 bit

SECONDA
GENERAZIONE 2: Il transistor è stato inventato nel 1948 da Shockley. In meno di 10 anni (1955-65) il transistor rivoluzionò i computer → quelli a valvole diventano obsoleti. Il primo calcolatore realizzato con i transistors fu chiamato TX-0 (Transistor-experimental computer 0)
↳ nel 1961 apparve il PDP-1 dotato di 4096 parole da 18 bit e poteva eseguire 200000 istruzioni al secondo → SPACEWAR primo videogioco
↳ PDP-8: più economica, nacque → omnibus → bus → è un insieme di cavi paralleli utilizzati per connettere i diversi componenti di un computer
↳ IBM 7094: diventava una delle principali forze del mercato
" 1401: leggeva e scriveva nastri magnetici e perforava schede e stampava output a un ritmo velocità

Calcolatori Elettronici

ARCHITETTURA DEI CALCOLATORI
CAP 1 2/



- CDC modee 6600 (Control Data Corporation) era ottima più veloce della 7094 venne chiamata number cruncher. Il segreto della velocità stava nel fatto che la CPU era al suo interno una macchina altamente parallela
  - Il progettista Cray continuò a realizzare calcolatori sempre più veloci: chiamati SUPERCOMPUTER

GENERAZIONE 3: - CIRCUITI INTEGRATI - L'invenzione dei circuiti integrati ('58) avvenuta da parte di Noyce permise di realizzare su un unico chip decine di transistors. Questo metodo assemblativo rese possibile la costruzione di computer più piccoli più veloci e più economici

- IBM System 360: lavoramento radicale → basata sui circuiti integrati e progettata si trattava di una famiglia di 6 macchine → e il SOFTWARE scritto per un processo modello funzionava senza problemi su uno più grande
  - MULTIPROGRAMMAZIONE: è possibile avere più programmi in memoria allo stesso tempo
  - emulazione: sui nuovi modelli si potevano emulare i vecchi senza perdere nulla
- DEC PDP-11: aveva i registri orientati alla pagina e la memoria orientata al byte

GENERAZIONE 4: - INTEGRAZIONE A GRANDISSIMA SCALA - Permise di inserire in unico chip decine di migliaia - migliaia di transistors → portò alla realizzazione dei microcomputer. Dal 1980 i prezzi crollarono talmente tanto che anche i privati → PC

- I personal computer venivano generalmente venduti in scatole di montaggio ovvero in KIT contenevano una scheda con un circuito stampato → inizialmente il SOFTWARE non era integrato ed era quindi compito dell'acquirente programmarlo
  - → Primo sistema Operativo CP/M scritto da Gary Kindall: girava sui floppy disk con un file system e una serie di istruzioni che l'utente doveva scrivere con la tastiera (CSHELL)
  - → Un altro dei primi personal COMPUTER fu l'APPLE progettati nel garage da Steve Jobs e Steve Wozniak
  - → L'IBM decise di scegliere una strada alternativa ovvero quella di non realizzare tutte le componenti da se ma di cercare di perfezionare quelle presenti e di assemblarle → il PC IBM diventa il computer più venduto nella storia
    - IBM inoltre decise di pubblicare e vendere tutti gli schemi del PC per poter rendere il modello fissato e quindi per permettere ad altre società di realizzare componenti aggiuntivi → non perse il suo scopo originale ma bensì alla nascita dei cloni PC

Le altre compagnie tra cui COMMODORE, APPLE e ATARI vennero schiacciate, l'unica che sopravvisse fu la MACINTOSH - APPLE → nel '84 venne realizzato il 1° PC dotato di una GUI (Graphical Interface "Interfaccia grafica per utente")

- Realizzazione dei primi PC PORTATILI: il primo fu l'OSBORNE 1 che con i suoi 11kg fu più che altro un computer bagaglio
  - qualche anno dopo COMPAQ produsse il suo primo clone PC e diventa leader nel mercato
- IBM e Microsoft volevano sviluppare MS-DOS diverso - operativo tuttavia vista la scarso successo → si creò una disputa tra Microsoft - IBM. Inseguito Microsoft e Intel riuscirono a detronizzare l'IBM

GENERAZIONE 5: Nel '82 il Giappone decise di dare vago alla 5° generazione di calcolatori - basati sull'intelligenza artificiale → progetto fallito

La quinta generazione vedeva comunque che le dimensioni dei PC diminuivano

→ APPLE NEWTON (PDA: personal Digital Assistant) migliorata l'interfaccia grafica

→ Weiser Obiquitas computing (computazione onnipresente)

TIPOLOGIE DI COMPUTER: (Moore co-fondatore della Intel)

LEGGE DI MOORE: Inizialmente stimata come previsione afferma che il numero di transistors raddoppia ogni 18 mesi → (up %60) all'interno di un chip

- La è questa legge ha creato il CICLO VIRTUOSO: avanzamenti tecnologici (transistor/chip) portano a prodotti migliori e allo stesso tempo più economici

LEGGE DI NATHAN: "il software è come un GAS", si espande fino a riempire il recipiente in cui è contenuto"

I computer nell'arco di un decennio diventano USA e GETTA

- **RFID**: più importante sviluppo (Radio Frequency Identification) → quando arrivano impulsi elettrici radio da un antenna esterna i chip RFID si accorgono grazie al segnale entrante sufficientemente lungo da riuscire a trasmettere all'antenna il proprio numero identificativo (a 128 bit)
  → I chip RFID si distinguono non solo per essere attivi o passivi ma anche in base allo spettro di radio-frequenze a cui possono rispondere.
  Quelli che operano a basse frequenze hanno una velocità di trasferimento dati limitata ma possono essere captati da un'antenna anche a grande distanza
  Quelli che operano ad alte frequenze hanno invece una più alta velocità di trasferimento dati, ma un raggio d'azione più ristretto
- **MICROCONTROLLORI**: computer integrati in un dispositivo, lo comandano e ne gestiscono l'interfaccia utente → sono incassati in vari dispositivi, tipicamente connessi in rete e si usano con modalità embedded
- **GAMES-COMPUTERS**: si trattano di normali computers dotati di speciali capacità grafiche e sonore, ma poco espandibili e forniti di SOFTWARE LIMITATO
  → PLAYSTATION 3: CPU IBM multicore a 128 bit a 3,2 GHz / 256 MB di ram
  → XBOX 360: chip grafico RSX da 256 MB a 550 MHz, CPU INTEL IBM XENON tricore a 32 bit – 512 MB di Ram chip grafico a 500 MHz (xenon)
- **SERVER**: Spesso vengono impiegate versioni potenziate dei personal computer o delle Workstation come server di rete.
  Dal punto di vista dell'architettura un server monoprocessore non è molto diverso da un personal computer mono-processore
- **CLUSTER**: Dato che il rapporto prezzo/prestazioni delle workstation continua progressivamente a migliorare i sistemisti hanno deciso di connettere fra loro un gran numero di queste macchine per formare i COW
  → queste macchine eseguono software speciale che permette loro di lavorare in modo congiunto su uno stesso problema, spesso di tipo scientifico o ingegneristico
  → quando si hanno centinaia o addirittura migliaia di Server → si parla di server FARM
- **MAINFRAME**: computer grandi come una stanza che nascono per i vecchi elaboratori ultimamente si stanno rinnovando ed ottimizzare grazie all'avvento di Internet
  → SUPERCOMPUTER: inizialmente erano ancora più potenti ed efficienti dei main frame, tuttavia con l'avvento dei COW hanno perso in questo settore

INTEL

PENTIUM IV: Nel '68 Moore – Rock – Noyce fondarono la INTEL CORPORATION
Nel 70' nacque il processore 4004, la prima CPU su un chip costituito da 2300 transistors
Successivamente durante la sua carriera produttiva, Intel aggiunse l'insieme d'istruzioni speciali MMX (Multimedia Extensions) per ottimizzare video/audio
Successivamente per migliorare le prestazioni con la grafica 3D furono aggiunte ulteriori istruzioni multimediali SSE

→ La versione del PIV a 3,06 GHz introdusse in una nuova e interessante caratteristica L'HYPERTHREADING: questa potenzialità permetteva ai programmi di dividere il loro lavoro in due flussi di controllo che potevano essere eseguiti in parallelo accelerandone l'esecuzione

Calcolatori Elettronici

ARCHITETTURA DEI CALCOLATORI
CAP 1 3/

INTEL:

CELERON: '98 una versione economica e con prestazioni ridotte pensata per i pC di fascia bassa

XEON: cache più grande, un bus più veloce e un miglior supporto multi processore

PENTIUM M: progettato per i computer portatili, questo chip faceva parte dell'architettura CENTRINO i cui obiettivi erano la riduzione del consumo energetico per prolungare la durata delle batterie

Tutti i processori sono retro-compatibili con i loro predecessori fino 8086 → Si è visto inoltre che la legge di Moore è valida anche per i processori

(1) Fino agli anni 2000 la strategia per migliorare le prestazioni del processore erano quelle di ~~migliorare~~ aumentare la frequenza di clock, tuttavia si capì ben presto che questa ~~no~~ via non era percorribile a causa del calore dissipato → grandi ventole per dissipare → rumore → fastidio utente

→ La via scelta per attualmente per aumentare la velocità di clock è quella di collocare più CPU sullo stesso CHIP → per via della relazione tra tensione e frequenza di clock (due CPU sullo stesso chip consumano molta meno potenza di una sola CPU con velocità doppia)

SPARC: Questo processore nacque da esigenze di avere un hardware specifico per sistemi operativi UNIX → nasce con la SUN-1
↳ Differentemente dalle altre macchine aveva workstation Sun era dotata di una connessione ethernet
↳ produttori fondarono nel 1982 la SUN MICROSYSTEMS

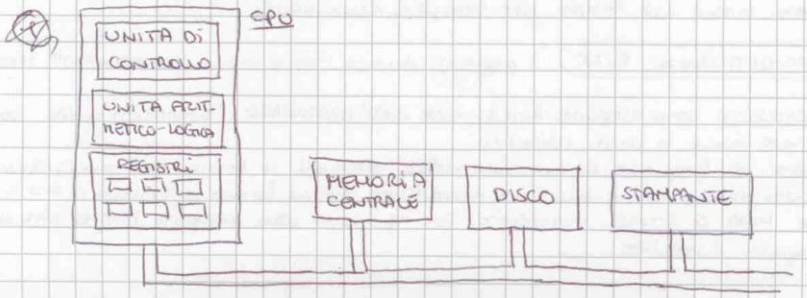
Nel '87 la SUN decise di progettare una propria CPU chiamata SPARC (Scalable Processor Architecture)

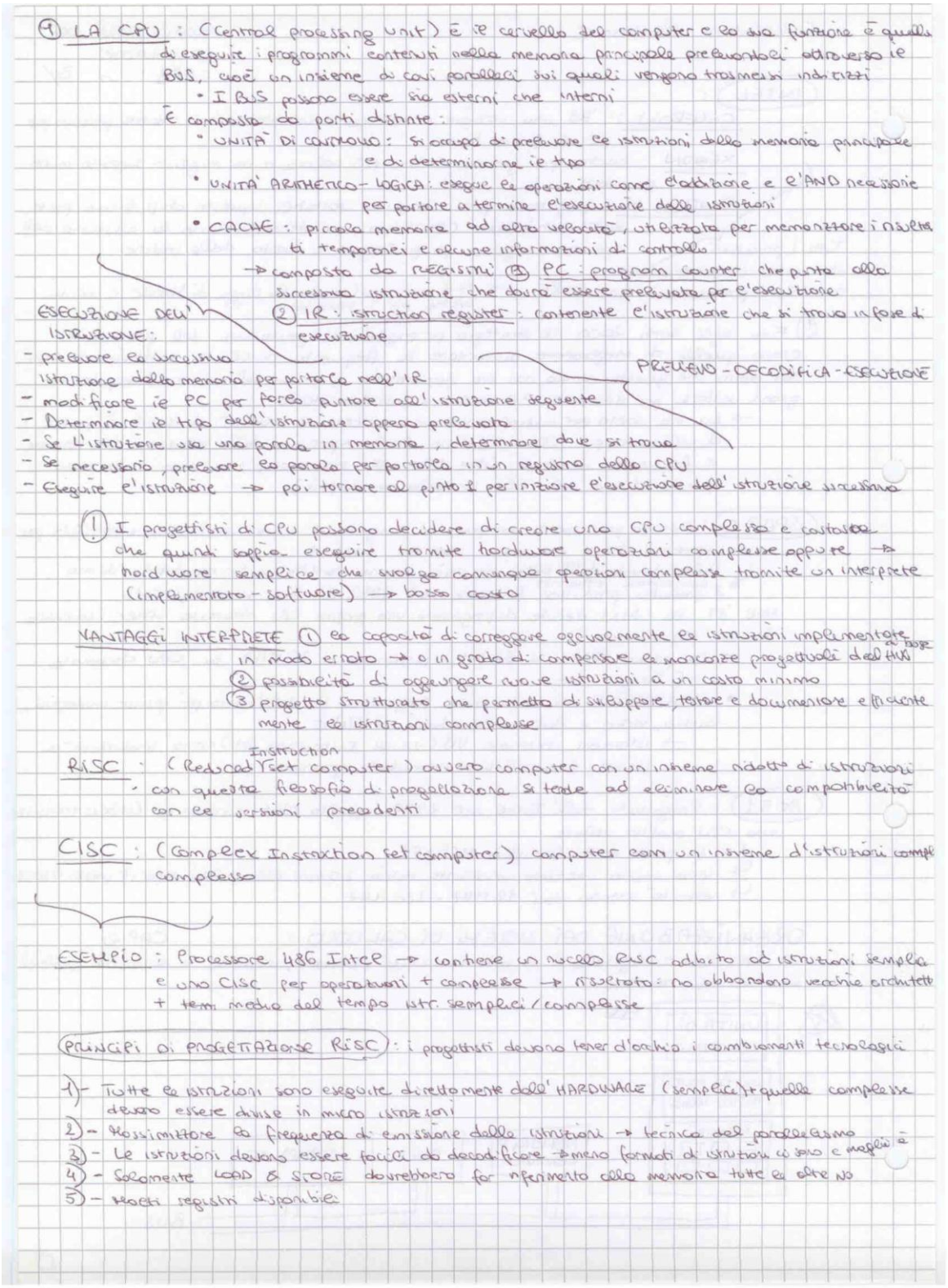
- La prima SPARC a 32 bit e funzionava a 36 MHz la CPU chiamata IU (Integer Unit
- ULTRA SPARC I era stata invece progettata fin dall'inizio per gestire immagini, audio, video e dati multimediali → 64bit
  → istruzioni chiamate VIS (Visual Instruction Set) erano finalizzate a fornire funzionalità multimediali generiche

8051: Progettato dall'Intel nel 1980, questo chip conteneva 60000 transistor una CPU molto veloce
↳ fa parte della famiglia MCS-51
↳ viene ancora utilizzato nonostante abbia 20 anni alle spalle → per il prezzo → 10cent
↳ velocità variano dai 12 MHz - 100 MHz

ORGANIZZAZIONE DEI SISTEMI DI CALCOLO: CAP 2

Un calcolatore digitale è un sistema in cui processori, memorie e dispositivi periferici sono connessi tra loro





① LA CPU : (Central processing unit) è il cervello del computer e la sua funzione è quella di eseguire i programmi contenuti nella memoria principale prelevandoli attraverso le BUS, cioè un insieme di cavi parallelli sui quali vengono trasmessi indirizzi
- I BUS possono essere sia esterni che interni

È composta da parti distinte:
- UNITÀ DI CONTROLLO: si occupa di prelevare le istruzioni dalla memoria principale e di determinarne il tipo
- UNITÀ ARITMETICO-LOGICA: esegue le operazioni come l'addizione e l'AND necessarie per portare a termine l'esecuzione delle istruzioni
- CACHE: piccola memoria ad alta velocità, utilizzata per memorizzare i risultati temporanei e alcune informazioni di controllo

→ composta da REGISTRI ① PC: program counter che punta alla successiva istruzione che dovrà essere prelevata per l'esecuzione

② IR: instruction register: contenente l'istruzione che si trova in fase di esecuzione

ESECUZIONE DELL'ISTRUZIONE:
- prelevare la successiva istruzione dalla memoria per portarla nell'IR
- modificare il PC per farlo puntare all'istruzione seguente
- Determinare il tipo dell'istruzione appena prelevata
- Se l'istruzione usa una parola in memoria, determinare dove si trova
- Se necessario, prelevare la parola per portarla in un registro della CPU
- Eseguire l'istruzione → poi tornare al punto 1 per iniziare l'esecuzione dell'istruzione successiva

PRELIEVO - DECODIFICA - ESECUZIONE

① I progettisti di CPU possono decidere di creare una CPU complessa e costosa che quindi sappia eseguire tramite hardware operazioni complesse oppure → hardware semplice che svolga comunque operazioni complesse tramite un interprete (implementato - software) → basso costo

VANTAGGI INTERPRETE: ① la capacità di correggere agevolmente le istruzioni implementate in modo errato → o in grado di compensare le mancanze progettuali dell'HW
② possibilità di aggiungere nuove istruzioni a un costo minimo
③ progetto strutturato che permetta di sviluppare, testare e documentare efficientemente le istruzioni complesse

RISC: (Reduced Instruction Set computer) ossia computer con un insieme ridotto di istruzioni
- con questa filosofia di progettazione si tende ad eliminare la compatibilità con le versioni precedenti

CISC: (Complex Instruction set computer) computer con un insieme d'istruzioni complesso

ESEMPIO: Processore 486 Intel → contiene un nucleo RISC addetto alle istruzioni semplici e uno CISC per operazioni + complesse → risultato: no abbandono vecchie architetture + tempo medio del tempo istr. semplici / complesse

PRINCIPI DI PROGETTAZIONE RISC: i progettisti devono tener d'occhio i cambiamenti tecnologici

1)- Tutte le istruzioni sono eseguite direttamente dall'HARDWARE (semplici) + quelle complesse devono essere divise in micro istruzioni
2)- Massimizzare la frequenza di emissione delle istruzioni → tecnica del parallelismo
3)- Le istruzioni devono essere facili da decodificare → meno formati di istruzioni ci sono e meglio è
4)- Solamente LOAD & STORE dovrebbero far riferimento alla memoria tutte le altre NO
5)- Molti registri disponibili

Calcolatori Elettronici

ARCHITETTURA DEI CALCOLATORI
CAP 2 4/

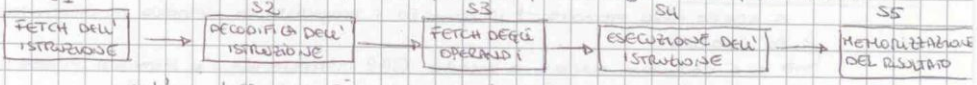
(1) LA CPU:

PARALLELISMO:

(1) - A LIVELLO DI ISTRUZIONE → è sfruttato all'interno delle singole istruzioni per far sì che la macchina possa elaborarne un maggior numero al secondo

(2) - A LIVELLO DI PROCESSORE - sono presenti più CPU che lavorano congiuntamente su uno stesso problema

→ PIPELINING: uno dei maggiori colli di bottiglia nella velocità di esecuzione delle istruzioni è rappresentato dal prelievo delle istruzioni dalla memoria
→ una tecnica proveniente era quella del prefetching ovvero divide l'esecuzione dell'istruzione in due parti
→ la pipelining esaspera questo concetto → suddivide in più parti che possono essere eseguite in parallelo, ognuna parte è eseguita da componenti hardware dedicati



→ L'uso della pipeline permette di bilanciare la LATENZA → tempo che un'istruzione impiega per essere elaborata LATENZA = nT dove n sta per i stadi T in nanosec
→ LARGHEZZA DI BANDA (MIPS) = 1000/T
→ VELOCITÀ di esecuzione di istruzioni al secondo = $10^9/T$

L'approccio di avere più pipeline cooperanti tra loro non risulta efficiente, quanto quello di associare più unità funzionali
→ ARCHITETTURE SUPERSCALARI: descrive processori che eseguono più istruzioni durante un ciclo di clock

△ ARRAY COMPUTER consiste di un gran numero di processori identici che eseguono la stessa sequenza di istruzioni su insiemi diversi di dati → in un array di processori le unità di elaborazione non sono delle CPU indipendenti, dato che c'è un'unica unità di controllo condivisa fra tutte
MULTIPROCESSORE: sistema composto da più CPU con una memoria comune. Dato che ogni CPU può leggere e scrivere una qualsiasi parte della memoria, esse devono coordinarsi (via software) per evitare di intralciarsi a vicenda
MULTICOMPUTER: visto che è difficile costruire multiprocessori con più di 256 → si è ovviato a questo problema collegando insieme più calcolatori sistema

(2) MEMORIA PRINCIPALE (RAM)

L'unità di base della memoria è il bit. Un bit può avere valore 0 o 1 ed è l'unità più semplice possibile
→ CELLE: possono memorizzare le informazioni, inoltre ciascuna cella ha un numero chiamato indirizzo → tutte le celle di una memoria contengono lo stesso numero di bit → ES: k bit → 2^k combinazioni di bit
SE UN INDIRIZZO ha m bit, il massimo numero di celle indirizzabili è 2^m
• la dimensione base è il byte che poi viene raggruppato in parole
↳ all'interno di una parola i byte possono essere numerati da sinistra a destra (BIG ENDIAN) oppure da destra a sinistra (LITTLE ENDIAN)

(CODICI CORRETTORI): occasionalmente le memorie dei calcolatori possono commettere degli errori per via di picchi di tensione sulla linea → si utilizzano per proteggersi dei codici di rilevazione/correzione → quando si utilizzano questi codici vengono aggiunti dei bit extra ad ogni parola di memoria secondo una modalità particolare

↳ PAROLA DI CODICE: $n = m + r$ dove m sta per input ed r per codice ridondante

DISTANZA di HAMMING: h = distanza di Hamming minima tra due codifiche valide del codice. Inoltre le proprietà di rilevazione e di correzione degli errori di una parola di codice dipendono dalla sua distanza di HAMMING

- RILEVARE: $h = d+1$
- CORREGGERE: $h = 2d+1$

(d = numero di bit diversi)

BIT DI PARITÀ: si aggiunge un singolo BIT di PARITÀ scelto in modo che il numero di bit nella parola di codice sia pari

↳ $(m + r + 1) \leq 2^r$ → fornisce un limite inferiore al numero di bit di controllo richiesti per correggere errori singoli

(m = bit dati, r = bit ridondanti)

- Tutti i bit la cui posizione è una potenza di 2 sono bit di parità; tutti quelli restanti sono usati invece per i dati
- Un modo per trovare i bit errati consiste nel calcolare inizialmente tutti i bit di parità. Successivamente si sommano in sommano tutti i bit di parità errati contando 1 per il bit 1, 2 per il bit 2, 4 per il bit 4 e via così e la somma risultante corrisponde alla posizione del bit errato

MEMORIA CACHE: Nel corso del tempo la squilibrio tra velocità del processore e velocità di accesso alla memoria → in quanto i progettisti di memoria ne incrementano solamente la capacità

↳ si è cercato di ovviare a questo GAP introducendo una memoria piccola e veloce chiamata CACHE (da ora cache che significa nascondere)

- Le parole di memoria usate più di frequente sono mantenute all'interno della cache
- quando la CPU necessita di una parola, la cerca nella cache, e solo nel caso in cui essa non sia presente la richiede alla memoria centrale

↳ **PRINCIPIO DI LOCALITÀ**: i riferimenti alla memoria fatti in un breve intervallo temporale tendono ad utilizzare solo una piccola frazione della memoria totale

Se durante un piccolo intervallo temporale una parola è letta o scritta K volte il calcolatore dovrà effettuare un solo riferimento alla memoria centrale e K-1 alla memoria veloce → maggiore è K, migliori sono le prestazioni complessive

↳ **HIT RATIO**: (frequenza di successi) che corrisponde alla frazione di riferimenti che può essere soddisfatta dalla CACHE $H = (K-1)/K$

- **TEMPO MEDIO DI ACCESSO** $= c + (1-h)m$ dove c sta per il tempo di accesso alla cache, m sta per il tempo di accesso alla memoria centrale

Le memorie centrali e la cache sono divise in blocchi di grandezza fissa per trarre vantaggio dal principio di località

→ Oggi si tende a favorire la cache specializzata, in cui le istruzioni sono memorizzate in una cache e i dati in un'altra → ARCHITETTURA HARVARD

Queste cache permettono che gli accessi avvengano in parallelo

ASSEMBLAGGIO MEMORIA: vari chip, in genere 8 o 16 elementi sono montati su una piccola scheda a circuiti stampati.

① SIMM (Single Inline Memory Module) a seconda che abbia i connettori allineati su uno o su due lati della scheda

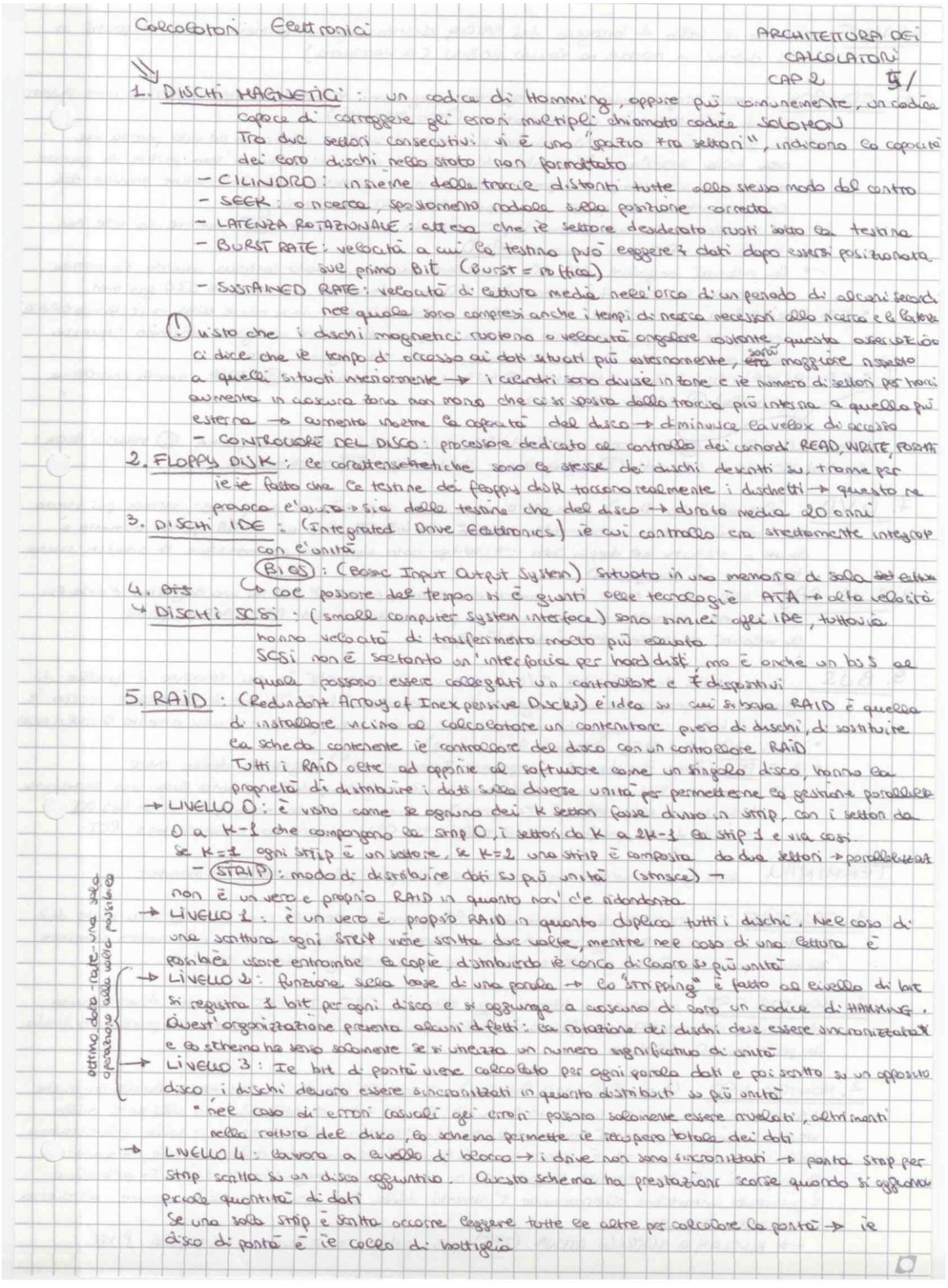
② DIMM (Dual Inline MemoryModule)

MEMORIA SECONDARIA

1. DISCHI MAGNETICI: consiste di uno o più piatti di alluminio rivestiti di materiale magnetico

TRACCIA: sequenza circolare di bit scritti mentre il disco compie una rotazione completa

SETTORI: suddivisioni delle tracce, contenenti in genere 512 byte di dati e pieni dati da un preambolo che permette alla testina di sincronizzarsi prima della lettura o della scrittura → poi segue un codice per la correzione di errore (ECC: Error Correction Code), un codice di Hamming, oppure più



Calcolatori Elettronici

ARCHITETTURA DEI CALCOLATORI
CAP 2 §1

1. DISCHI MAGNETICI: un codice di Hamming, oppure più comunemente, un codice capace di correggere gli errori multipli: chiamato codice Solomon
Tra due settori consecutivi vi è uno "spazio tra settori", indicano la capacità dei loro dischi nello stato non formattato
- CILINDRO: insieme delle tracce distanti tutte allo stesso modo dal centro
- SEEK: o ricerca, spostamento radiale alla posizione corretta
- LATENZA ROTAZIONALE: attesa che il settore desiderato ruoti sotto la testina
- BURST RATE: velocità a cui la testina può leggere i dati dopo essersi posizionata sul primo bit (Burst = raffica)
- SUSTAINED RATE: velocità di lettura media nell'arco di un periodo di alcuni secondi nel quale sono compresi anche i tempi di ricerca necessari alla ricerca e alla lettura

(!) visto che i dischi magnetici ruotano a velocità costante durante questa esecuzione ci dice che il tempo di accesso ai dati situati più esternamente, sarà maggiore rispetto a quelli situati internamente → i cilindri sono divise in tante e il numero di settori per traccia aumenta in corsa sono non man che ci si sposta dalla traccia più interna a quella più esterna → aumenta mentre la capacità del disco → diminuisce la velocità di accesso
- CONTROLLORE DEL DISCO: processore dedicato al controllo dei comandi READ, WRITE, FORMAT

2. FLOPPY DISK: le caratteristiche sono le stesse dei dischi descritti su, tranne per il fatto che le testine dei floppy disk toccano realmente i dischetti → questo ne provoca l'usura → sia della testina che del disco → durata media 20 anni

3. DISCHI IDE: (Integrated Drive Electronics) il cui controllo è direttamente integrato con l'unità
(BIOS): (Basic Input Output System) situato in una memoria di sola lettura
Le sue possibilità è quindi legate alla tecnologia ATA → alta velocità

4. ~~DIS~~

5. DISCHI SCSI: (small computer system interface) sono simili agli IDE, tuttavia hanno velocità di trasferimento molto più elevata
SCSI non è soltanto un'interfaccia per hard disk, ma è anche un bus al quale possono essere collegati un controllore e 7 dispositivi

6. RAID: (Redundant Array of Inexpensive Disks) l'idea su cui si basa RAID è quella di installare vicino al calcolatore un contenitore pieno di dischi, di sostituire la scheda contenente il controllore del disco con un controllore RAID
Tutti i RAID oltre ad apparire al software come un singolo disco, hanno la proprietà di distribuire i dati sulle diverse unità per permettere la gestione parallela
→ LIVELLO 0: è visto come se ognuno dei K settori fosse diviso in strip, con i settori da 0 a K-1 che compongono la strip 0, i settori da K a 2K-1 la strip 1 e via così.
Se K=1 ogni strip è un settore, se K=2 una strip è composta da due settori → parallelismo
- (STRIP): modo di distribuire dati su più unità (stripe) →
non è un vero e proprio RAID in quanto non c'è ridondanza
→ LIVELLO 1: è un vero e proprio RAID in quanto duplica tutti i dischi. Nel caso di una scrittura ogni STRIP viene scritta due volte, mentre nel caso di una lettura è possibile usare entrambe le copie: il carico di lavoro su più unità
→ LIVELLO 2: funziona sulla base di una parola → lo "striping" è fatto al livello di bit si registra 1 bit per ogni disco e si aggiunge a ciascuno di loro un codice di HAMMING. Quest'organizzazione presenta alcuni difetti: la rotazione dei dischi deve essere sincronizzata e lo schema ha senso solamente se si usano un numero significativo di unità
→ LIVELLO 3: Il bit di parità viene calcolato per ogni parola dati e poi scritto su un apposito disco; i dischi devono essere sincronizzati in quanto distribuiti su più unità
- nel caso di errori casuali gli errori possono solamente essere mascherati, altrimenti nella ricerca del disco, lo schema permette il recupero totale dei dati
→ LIVELLO 4: lavora a livello di blocco → i drive non sono sincronizzati → porta strip per strip scritta su un disco aggiuntivo. Questo schema ha prestazioni scarse quando si aggiorna piccole quantità di dati
Se una sola strip è scritta occorre leggere tutte le altre per calcolare la parità → il disco di parità è il collo di bottiglia

ottimo dato - rate una sola operazione alla volta possibile

→ RAID 5 : elimina il collo di bottiglia del RAID 4, distribuendo uniformemente i bit di parità su tutti i dischi in modalità ROUND ROBIN (a scacchiera)

6. CD-ROM : creati originariamente per registrare programmi televisivi → oggi utilizzati come dispositivi di memorizzazione per calcolatori (700/750 MB)
→ La preparazione di un CD avviene per mezzo di un laser infrarosso ad alta potenza che crea sulla superficie fotosensibile di un disco di vetro dei buchi, a partire da questo viene creato uno stampo che presenta rilievi in corrispondenza delle incisioni prodotte dal laser, sul quale si inserisce del policarbonato liquido.
Le incisioni nel substrato di policarbonato sono chiamate PIT mentre le aree non incise tra i PIT sono chiamate LAND
→ La velocità angolare del CD deve essere ridotta, in modo continuo, quando la testina di lettura si sposta dal centro del CD verso l'esterno → relax → 530 giri min
• CD SCRIVIBILI: per scrivere su un CD la potenza del laser deve essere alta → il fascio colpisce una regione del pigmento, esso la scalda al punto da rompere un legame chimico e questo cambiamento della struttura molecolare crea una regione scura.
• CD-RISCRIVIBILI: laser a tre potenze ① ALTA: cristallino - amorfo ② MEDIA: amorfo cristallino (cancella) ③ BASSA: legge i dati

STANDARD CD: ① RED BOOK: standard ISO per i CD audio (simboli e forme) ② YELLOW BOOK: standard ISO per i CD dati ③ GREEN BOOK: standard per CD multimediali

7. DVD : (Digital Video Disk) notevole aumento della capacità, pit più piccoli, spirale più stretta e dimensioni uguali ~~in base alla capacità~~ da 4,7 GB fino ad 17 GB questo aumento è dovuto a scrittura su doppio strato → impiego noto soprattutto per soddisfare le richieste cinematografiche di HOLLYWOOD.

8. BLUE-RAY: utilizza un laser blu, invece che rosso, poiché il blu ha una lunghezza d'onda più piccola che permette una messa a FUOCO più accurata e quindi PIT e LAND più piccoli
La velocità di trasferimento è di 4,5 MB/s

9. BUS : quasi tutte le periferiche I/O hanno un CONTROLLORE il cui compito è quello di governare il proprio dispositivo, e gestire il suo accesso al bus. Inoltre il controllore spezza il flusso di bit in unità, generalmente costituite da uno o più parole e successivamente le scrive in memoria
• BUS ISA (Industry Standard Architecture)
• BUS PCI (Peripheral Component Interconnect") progettato dalla Intel
→ in questo BUS comunica direttamente con il controllore della memoria lungo una connessione dedicata ad alta velocità. Il controllore comunica con la memoria e con il bus PCI in modo diretto, per far sì che il traffico tra CPU e memoria non occupi il bus PCI

TERMINALI : sono generalmente composti da due parti principali: tastiera e monitor

1. TASTIERE : la pressione di un tasto di un PC genera un interrupt che stimola una porta del sistema operativo → leggendo un registro hardware della tastiera ricava il numero associato al tasto premuto. Anche quando si rilascia un tasto viene generato un interrupt

2. MONITOR CRT: contiene un tubo a raggi catodici CRT. Il CRT è dotato di un cannone elettronico che indirizza un fascio di elettroni contro uno schermo fosforescente posizionato vicino la parte frontale del tubo

2. MONITOR LCD : (Liquid Crystal Display) I cristalli liquidi sono ~~sostanze~~ molecole organiche viscose che si muovono come un liquido. Quanto tutte le molecole sono allineate nella stessa direzione le proprietà ottiche del cristallo dipendono dalla direzione e dalla polarizzazione della luce incidente → è possibile modificare l'allineamento delle molecole, e quindi le proprietà ottiche, tramite l'applicazione di un campo ~~magnetico~~ elettrico. In particolare è possibile controllare elettronicamente l'intensità della luce uscente facendola passare attraverso un cristallo liquido
→ DISPLAY A MATRICE ATTIVA (TFT) più costosi e si occupa del singolo Pixel

Calcolatori Elettronici

ARCHITETTURA DEI CALCOLATORI
CAP 2 6/

MONITOR: (VIDEO RAM): memoria dedicata alla gestione video → situata nel control core video. La sua dimensione dipende dalla sua risoluzione.
L'immagine è costituita da una matrice di punti (pixel) dimensione PxNxK
P: n° di pixel; N: n° di byte per pixel; K n° di matrici
→ Intel per permettere una larghezza di banda maggiore tra la CPU e la RAM delle schede VIDEO ha cominciato a impiegato un nuovo bus verso la RAM video chiamato bus AGP (accelerated graphic port)

MOUSE: unità come strumento per puntare sullo schermo. Sono stati prodotti 3 tipi di mouse ottici, meccanici, opto-meccanici. Quelli più utilizzati sono quelli a LED (Light Emitting Diode) quando il mouse si muove il fotorivelatore riconosce l'incrocio tra due linee rilevando un cambiamento nella quantità di luce generata dal LED che viene riflessa.

STAMPANTI: ① A GETTO di INCHIOSTRO: sono i mezzo favorito per lo stampa casalinga a basso costo. Una testina mobile, contenente una cartuccia di inchiostro, scorre orizzontalmente lungo il foglio per mezzo di un motore, mentre l'inchiostro viene spruzzato da piccoli ugelli
② STAMPANTE LASER: Il cuore della stampante è un tamburo rotante molto preciso, è ricoperto con un materiale fotosensibile. La luce generata dal laser passa per tutta la lunghezza del tamburo → per ottenere la deviazione orizzontale si utilizza uno specchio ottagonale → il fascio luminoso viene modulato per produrre regioni luminose e regioni scure. Il tamburo dopo aver raggiunto disegnato una linea di punti, ruota → questo linea raggiunge il toner che viene attirato dai punti con una carica elettrica. Il tamburo ricoperto di toner viene premuto contro il foglio di carta, trasferendo la polvere nera.
③ STAMPANTI A COLORI: presentano una serie di complicazioni, come ad esempio il fatto che lo schermo utilizza RGB (a 3 colori) mentre la stampante (CMYK) (Yellow, Magenta, Black, Ciano)

APPARECCHIATURE PER LE TELECOMUNICAZIONI

① MODEM: dispositivo che riceve da un calcolatore i caratteri sotto forma di segnali a due livelli, un bit alla volta e che trasmette i bit in gruppi di uno o due utilizzando una modulazione d'ampiezza di frequenza o fase
- quasi tutti i modem sono FULL-DUPLEX ie che significa che possono trasmettere allo stesso tempo sia in UPLINK che in DOWNLINK
- solitamente la banda offerta dal doppino telefonico viene divisa in canali ie primo è quello dedicato al POTS (Plain Old Telephone Service)
- ADSL processore di segnale digitale impostato opportunamente per funzionare come 250 modem in parallelo a diverse frequenze

② MACCHINE FOTOGRAFICHE DIGITALI: In una macchina fotografica digitale funziona nello stesso modo delle vecchie fotocamere → in + ha una griglia rettangolare di CCD (Charge-Coupled Device). Quando questo dispositivo è colpito dalla luce si carica elettricamente, in modo proporzionale alla quantità di luce ricevuta
→ si utilizzano 4 CCD per ogni pixel al posto di tre perché l'occhio umano è più sensibile alla luce verde che alle luci rosse e blu
Es: MACCHINETTA A 6M → non è realmente così in quanto prendono in considerazione i CCD e quindi 6M → reali: 1,5M
→ quando viene premuto il pulsante per scattare una foto il software compie tre azioni ① Imposta la messa a fuoco ② Determina l'esposizione e calcola il bilanciamento del bianco

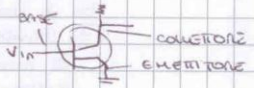
CODIFICHE DEI CARATTERI: Ogni calcolatore ha un insieme di caratteri indispensabili che comprendono 26 lettere maiuscole, minuscoli e vari caratteri speciali. Per poter utilizzare questi caratteri nel calcolatore occorre assegnare loro un numero
- La corrispondenza tra caratteri e numeri naturali costituisce un CODICE DI CARATTERI

1. ~~ASCII~~ ASCII: un codice ampiamente utilizzato è ASCII (American Standard Code for Information Interchange". Questo tipo di caratteri sono definiti da 8 bit permettendo così di avere una tabella di 128 caratteri distinti

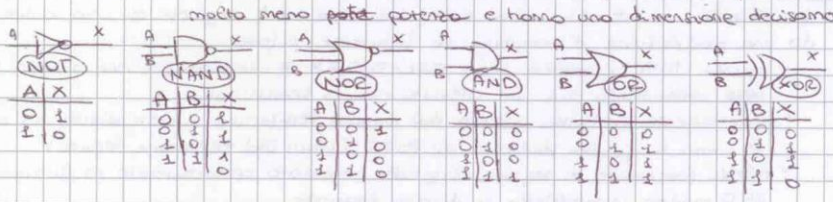
2. UNICODE: Standard Internazionale è supportato da alcuni linguaggi di programmazione l'idea alla base di UNICODE consiste nell'assegnare a ogni carattere un valore a 16 bit, chiamato CODE POINT → non esistono quindi caratteri o sequenze speciali composte da più byte → rende più semplice la scrittura di programmi → L'unicode risolve moltissimi problemi ma non tutti
   - PROBLEMI: Non tutti gli alfabeti sono messi nell'ordine del dizionario internazionale inoltre i code point sono prefissati per determinate lingue → se si introducono nuove parole (ES: Grappolone) si dovrebbero aggiungere nuovi code point

LIVELLO LOGICO DIGITALE CAP 3

I circuiti digitali possono essere costruiti combinando tra loro un piccolo numero di componenti elementari. CIRCUITO DIGITALE: è un circuito in cui sono presenti solo due valori logici: (1) un segnale da 2 a 5 Volt rappr. num binario 1 (2) un segnale da 0 a 1 volt rappr. numero 0
- PORTE LOGICHE: piccoli dispositivi elettronici → ciascuno delle quali calcola una diversa funzione di segnali
- TRANSISTOR: tutta la logica digitale si fonda sul fatto che un transistor può essere costruito in modo da funzionare come un velocissimo interruttore binario
  - hanno 3 connessioni verso l'esterno: il collettore, la base e l'emettitore
  - nei calcolatori si utilizzano porte NAND e NOR in quanto necessitano di 2 transistor invece dei 3 utilizzati per le porte OR & AND normali



  - Principali tecniche di costruzione delle porte logiche
    - BIPOLARE: TTL (transistor-transistor-logic) → motore dell'elettronica digitale
      ECL (Emitter-Coupled Logic) → quando è richiesto un funzionamento a velocità molto elevate
    - MOS (Metal Oxide Semiconductor) sono più lente di quelle bipolari ma richiedono molto meno ~~potenza~~ potenza e hanno una dimensione decisamente inferiore



A	X
0	1
1	0

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

ALGEBRA BOOLEANA: ottima per descrivere i circuiti combinatori e porte logiche in cui variabili e funzioni possono assumere soltanto i valori 0 e 1
- una funzione booleana di n variabili può essere completamente descritta con una tabella con 2^n righe → dato che esistono 2^n possibili combinazioni dei valori input → TABELLA DI VERITÀ
- visto che al crescere degli ingressi aumenta in modo esponenziale si è soliti prendere dalla TABELLA di VERITÀ solamente le ~~non~~ i valori 1 → ES: $\bar{A}.B + A.\bar{B}$ (+ = OR)

- un circuito elettronico può implementare una funzione booleana che utilizza segnali per rappresentare le variabili di input e di output oltre alle classiche porte logiche
- IMPLEMENTARE UN CIRCUITO CHE REALIZZI UNA QUALSIASI FUNZIONE BOOLEANA
  - Si scrive la tabella di verità della funzione
  - Ci si munisce di invertitori per generare la negazione di ciascun input
  - Si utilizza una porta AND per ciascun termine le cui valore nella colonna risultato vale 1
  - Si collegano le porte AND agli input appropriati
  - Si connettono tutti gli ~~input~~ output delle porte AND nella porta OR

Calcolatori Elettronici

ARCHITETTURA DEI CALCOLATORI CAP3 7/

LIVELLO LOGICO DIGITALE

Sia le porte NAND che le porte NOR costituiscono un insieme di connettivi FUNZIONALMENTE COMPLETO, nel senso che una qualsiasi funzione booleana può essere calcolata usando soltanto uno di questi due tipi di porte

PROPRIETÀ ALGEBRA BOOLEANA: ① proprietà distributiva, è possibile fattorizzare AB+AC nella forma A(B+C) ② DUALITÀ: scambiando AND con OR e anche 1 con 0 è possibile creare una delle due forme a partire dall'altra

→ una stessa porta logica reale può calcolare diverse funzioni a seconda delle convenzioni adottate (a) CONVENZIONE LOGICA POSITIVA: secondo la quale 0 volt è il valore logico 0 e 5 volt è il valore logico 1 (b) LOGICA NEGATIVA: 0 volt rappresenta il valore logico 1 e 5 volt il valore logico 0 → la convenzione scelta per assegnare le tensioni ai valori logici riveste un importante cruciale

CIRCUITI INTEGRATI: le porte logiche sono prodotte e vendute in sottoforma di circuiti integrati → ai quali ci si riferisce con IC: ovvero quadrato di silicio 5×5 sul quale sono state fabbricate alcune porte, DIP (Dual Inline Package) ci si riferisce al contenitore che all'esterno presenta due file di PIN. Il CHIP: è la somma di questi due componenti

- CIRCUITI SSI (Small Scale Integrated): da 1 a 10 porte
- CIRCUITI MSI (Medium Scale Integrated): da 10 a 100 porte
- CIRCUITI LSI (Large Scale Integrated): da 100 a 100.000 porte
- CIRCUITI VLSI (Very Large Scale Integrated): più di 100.000 porte

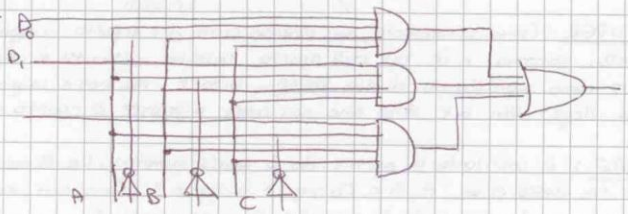
RITARDO DELLA PORTA: comprende sia il tempo dovuto alla propagazione del segnale attraverso il chip, sia il tempo di commutazione (intorno compresi tra 1 e 10ns)

SSI: ~~contiene in genere tra le due e le 5 porte logiche~~

RETI COMBINATORIE: contengono più input e più output, in cui gli output sono univocamente determinati dagli input

↳ MULTIPLEXER: è un circuito con 2^n dati di input, un valore di output e n input di controllo, gli input di controllo permettono di selezionare uno dei dati di input che viene "instradato" verso l'output

ES



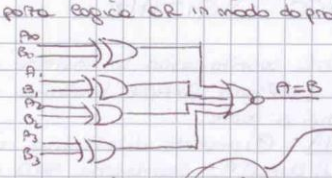
- una applicazione reale del MULTIPLEXER è la conversione di dati da parallelo a seriale

↳ DE-MULTIPLEXER: redirige il suo segnale di input, verso uno dei 2^n output in base ai valori delle linee di controllo, se il valore binario definito dalle linee di controllo è k, viene selezionato l'output k

↳ DECODIFICATORI: (decoder) accetta come input un numero di n bit e lo utilizza per impostare a 1 una sola delle 2^n linee di output

ES: Fornisco alla memoria un indirizzo, e in base a questo utilizzo i suoi 3 bit + significativi (ABC) per far sì che solo una delle linee di OUTPUT assuma valore 1 mentre le altre 0. Ciascuna linea di output abiliterà un determinato chip Quindi la linea che assumerà valore 1, abiliterà il suo chip

COMPARATORI: permette di confrontare due stringhe di bit. Se le due stringhe in ingresso sono uguali, tutte e 4 le porte XOR devono generare come risultato. Questi 4 segnali possono poi essere connessi a una stessa porta logica OR in modo da produrre un valore 0 quando gli input sono uguali ed 1 in caso contrario



chip molto generale che permette di calcolare somme di prodotti
- permette di realizz. una qualsiasi funzione booleana

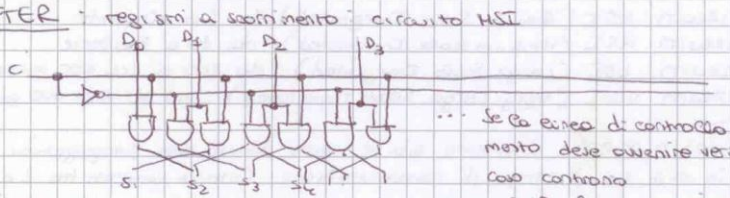
ARRAY LOGICI PROGRAMMABILI: (PLA) Il cuore del circuito è costituito da uno schiera di 50 porte AND che possono avere come input un qualsiasi sottoinsieme dei 24 segnali di input. Ogni linea di input connessa alle 50 porte AND contiene un fusibile. Al momento della fabbricazione del CHIP i 1200 fusibili sono intatti. Successivamente l'utente può programmare la matrice bruciando i fusibili con l'applicazione di un alta tensione.

↳ l'uscita del circuito consiste in 8 porte OR, che possono avere fino a 50 input corrispondenti agli output delle porte AND.

↳ il CHIP ha 20 pin in tutto, 12 per gli input, 8 per gli output, e 2 per la tensione e la terra

CIRCUITI PER L'ARITHETICA

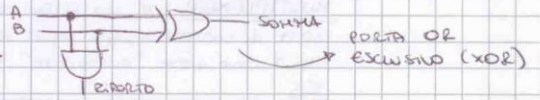
SHIFTER: registri a scorrimento: circuito MSI



... se la linea di controllo C ha valore 0 lo spostamento deve avvenire verso sinistra, e 1 in caso contrario

OSS: per tutti i bit c'è una ~~sola~~ coppia di porte AND, fatta eccezione per le porte che si trovano alle estremità. Quando C=1 la porta che si trova a destra in ciascuna coppia viene abilitata (escludo possibile il bit corrispondente verso l'output dato che la porta AND è collegata all'input della porta OR alla sua destra si ottiene uno scorrimento verso destra. Quando C=0 è la porta AND di sinistra in ciascuna coppia a essere abilitata, producendo uno spostamento verso sinistra

SOMMATORI:



↳ **HALF-ADDER**: (semisommatore, non tenendo conto del riporto in ingresso) capace di calcolare il bit della somma e il bit del riporto. Questo circuito è in grado di sommare correttamente i bit meno significativi di due stringhe binarie, ma non è in grado di eseguire correttamente la somma degli altri bit, dato che non tiene a genere il riporto che arriva dalle posizioni precedenti

↳ **SOMMATORE**: è costituito a partire da 2 semisommatori. La linea di output Sum vale 1 se una o tre delle linee A, B e Carry in valgono 1. Carry out vale 1 se sia A sia B valgono 1 oppure se uno dei due vale 1 e anche il carry in vale 1
- l'unione dei due semisommatori permette di generare in output, sia il bit della somma sia il bit del riporto

↳ **SOMMATORE A PROPAGAZIONE DI RIPORTO**: Per un dato bit il riporto in uscita viene trattato come riporto in entrata per il bit alla sua sinistra

↳ **SOMMATORE A SELEZIONE DI RIPORTO**: invece di avere un singolo sommatore per i bit più significativi, ne creiamo due identici in parallelo duplicando l'HARDWARE corrispondente → il circuito consiste in 3 sommatori a 16 bit, uno per i bit meno significativi e due per quelli più significativi, funzionanti in parallelo
- di mezzo tempo del tempo dedicato all'addizione

Calcolatori Elettronici — ARCHITETTURA DEI CALCOLATORI — CAP3 8/

UNITÀ ARITMETICO LOGICHE (ALU)

può calcolare una qualsiasi delle 4 funzioni, AND, OR, somma A+B in base al valore binario definito dalle linee F_0 e F_1, preposte alla selezione della funzione matematica

- SETTORE IN BASSO A SINISTRA: contiene un decodificatore a 2 bit, che permette di generare in base ai valori di F_0 e F_1 i segnali di attivazione delle 4 operazioni
- SETTORE IN ALTO A SINISTRA: contiene le porte logiche necessarie per calcolare A AND B, A OR B, e $\bar{B}$ → in base alle linee di attivazione che escono dal decodificatore, una solo di questi risultati viene passato alla porta finale OR con le uscite 3 le altre vengono tutte 0
- SETTORE IN BASSO A DESTRA: contiene un sommatore per calcolare la somma di A e B e a gestire allo stesso tempo i riporti → importante poiché è necessario dato che con ogni probabilità vari circuiti dello stesso tipo potrebbero essere collegati fra loro per permettere operazioni su interi parole (BIT SLICES)

↳ CLOCK - utile per gestire la sincronizzazione e permettere ai progettisti di ottenere relazioni temporali desiderate

→ è un circuito che emette una serie di impulsi di larghezza definito a intervalli temporali costanti. L'intervallo temporale tra due impulsi consecutivi → ciclo di clock

→ visto che possono verificarsi più eventi durante lo stesso ciclo, si è soliti suddividere i cicli di clock in sottocicli

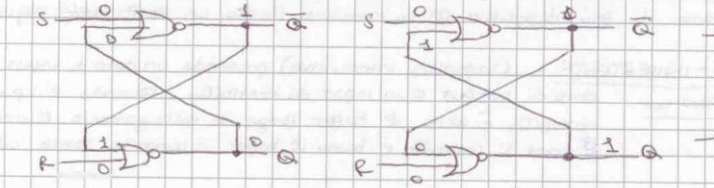
- si intercetta la linea di clock principale e si inserisce in un circuito di cui si conosce i ritardo → in questo modo è possibile generare un secondo segnale di clock, la cui fase è traslata rispetto a quello del clock principale

→ i clock sono SIMMETRICI, nel senso che il tempo speso nello stato in cui il segnale è alto è uguale al tempo speso quando il segnale è basso

- per generare una sequenza di impulsi asimmetrici, il clock principale viene sfasato utilizzando un circuito ritardante e ne viene calcolato l'AND logico rispetto al segnale

MEMORIA: utilizzato per conservare sia le istruzioni da eseguire sia i dati

↳ LATCH: ha due input S, setting (il valore del LATCH è 1 per azzerarlo (Resetting) il circuito ha anche due OUTPUT, Q e $\bar{Q}$ che sono complementari



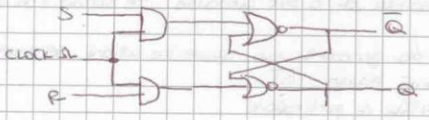
→ Per S = R = 0, il latch ha due stati stabili, 0, 1 in base al valore di Q

→ si può dire che quando S è impostato temporaneamente a 1 lo stato del latch diventa Q = 1 indipendentemente dallo stato in cui si trovava precedentemente

→ allo stesso modo quando si imposta temporaneamente R a 1 si forza il LATCH a portarsi nello stato Q = 0

- Il circuito accetta quindi quale valore, se S o R è stato settato per primo

↳ LATCH SR TEMPORIZZATO: non cambia lo stato in specifici momenti



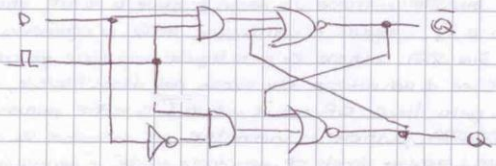
→ quando il clock vale 0 (generalmente) entrambe le porte AND generano in OUTPUT 0 indipendentemente dai valori di S e R, impedendo quindi al LATCH di cambiare stato

→ Quando il clock vale 1 le porte AND non bloccano più i segnali S e R e possono tornare a puntare lo stato del LATCH

→ I termini ENABLE e STROBE sono spesso utilizzati per indicare che l'input clock vale 1, ovvero che il circuito è indipendente dallo stato di S e da quello di R

LATCH D TEMPORIZZATO : un modo per risolvere l'ambiguità del latch SR causato dalla situazione S=R=1 è dato dal fatto di considerare un solo input D, quando il clock vale 1 il valore corrente di D viene campionato e memorizzato nel latch. Questo circuito, chiamato LATCH D temporizzato è una vera e propria memoria a 1 bit, in cui il valore memorizzato è sempre disponibile sulla uscita Q

- questo circuito prevede 11 transistor

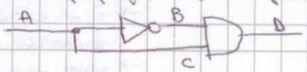


→ un LATCH è a commutazione di LIVELLO

FLIP-FLOP : circuiti per campionare il valore di una certa area in un particolare istante e memorizzarlo. La transizione di stato non si verifica quando il clock vale 1, ma durante la transizione del clock da 0 a 1 oppure da 1 a 0

- commutato sul fronte

→ **INVERTITORE** : indica un piccolo ma non nullo ritardo di propagazione che permette al circuito di funzionare in modo corretto →

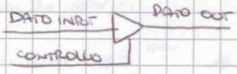


Quando gli input b e c vengono processati dalla porta AND si ottiene come risultato un impulso di breve durata, la cui larghezza Δ dell'impulso è uguale al ritardo dell'invertitore. L'output della porta AND corrisponde semplicemente a questo impulso sfasato del ritardo interno alla porta. Questo sfasamento temporale significa che il latch D verrà attivato con un ritardo fisso rispetto al fronte di salita del clock → non ha alcun effetto sulla durata dell'impulso

- Molti LATCH e FLIP-FLOP, seppur non tutti hanno anche l'output $\bar{Q}$, e alcuni sono dotati di due input aggiuntivi, SET o PRESET (che forzano lo stato a Q=1) e Reset o clear (che forzano lo stato a Q=0)

REGISTRI : L'unione di più dispositivi per la memoria forma un ~~FLIP-FLOP~~ registro

→ **BUFFER NON-INVERTENTE** : (semplice interruttore) possiede un dato di input, un dato di output e un input di controllo. Quando l'input di controllo è alto, il buffer funge da collegamento. Quando invece l'input di controllo è basso il buffer si comporta come circuito aperto



→ **BUFFER INVERTENTI** : si comporta come un normale invertitore, che si comporta come un normale invertitore quando il controllo ha valore alto e disconnette l'output dal circuito quando il controllo ha valore basso

↳ I buffer inoltre amplificano il segnale e possono quindi guidare più input allo stesso tempo

I BUFFER sono dispositivi a 3 stati, in quanto possono generare in out: 1, 0, circuito aperto

CHIP DI MEMORIA : → 4×3 memoria a 4 parole di 3 bit ciascuna → hanno una struttura interna, con schema bidimensionale ripetuto

- SEGNALE ASSERITO : è impostato in modo da generare una qualche azione ESEMPI
  CS: ASSERITO VALORE ALTO CS / VALORE BASSO: $\overline{CS}$
- SEGNALE NEGATO : quando non succede nulla di particolare

Dato che un calcolatore ha di solito vari chip di memoria è necessario disporre di un segnale che selezioni il chip richiesto in un dato momento, in modo che solo esso possa rispondere al segnale mentre tutti gli altri lo ignorano

Calcolatori Elettronici

ARCHITETTURA DEI CALCOLATORI
CAP 3 9/

CHIP DI MEMORIA: $\overline{WE}$ (Writing Enable); $\overline{OE}$ (Output Enable)

ES: MEMORIA 4Mb → al suo interno il chip è organizzato come una matrice 2048 x 2048 celle ad 1 bit. Per indirizzare il chip si seleziona inizialmente una riga, immettendo un numero a 11 bit sui bit dell'indirizzo e asserendo il segnale $\overline{RAS}$ (Row Address Strobe). Successivamente si immette sui pin dell'indirizzo un numero di colonna e si asserisce il segnale $\overline{CAS}$ (Column Address Strobe)

- i chip delle grandi memorie sono costruiti come matrici n x n, indirizzate da numeri di riga e colonna → questo tipo di architettura indica il numero di pin ma rende il chip più lento in quanto sono necessari due cicli di indirizzamento, uno per la riga e uno per la colonna
  → per guadagnare un po' di velocità → si specifica un indirizzo di riga seguito da una sequenza di indirizzi di colonna, in modo da poter accedere a bit consecutivi all'interno della stessa riga

ES:

RAM (Random Access Memory) memorie che possono essere sia lette che scritte. Esistono due tipi di RAM

① - SRAM: sono statiche e sono costruite utilizzando circuiti simili ai flip-flop D e hanno la proprietà di mantenere, il proprio contenuto, fintanto che vi è alimentazione → sono molto veloci (nsec) e sono utilizzate come memorie CACHE di secondo livello

② RAM-DINAMICHE: hanno un elevata densità (molti bit per chip) richiedono un transistor e un condensatore per bit → hanno grande capacità → bassa velocità
→ DRAM FPM (Fast Page mode) utilizzato nei calcolatori più vecchi. Internamente è organizzato a matrice di bit, per funzionare occorre che l'hardware fornisca prima l'indirizzo di riga e poi quello di colonna. Il funzionamento della memoria è asincrono rispetto al clock principale del sistema

→ DRAM EDO (Extended Data Output): il riferimento alla memoria può avere inizio ancor prima che sia completato il precedente → aumenta la larghezza di banda della memoria

→ SD RAM (Synchronous DRAM): ibrido, in parte statico ed in parte dinamico ed è guidato dal clock principale del sistema. Il clock elimina la necessità dei segnali di controllo, per specificare al chip quando deve rispondere
- l'eliminazione dei segnali di controllo aumenta il tasso di trasferimento dati tra CPU e memoria

→ SD RAM DDR: (double data Rate) In queste memorie il chip produce un output sul fronte di salita del segnale di clock e uno sul fronte di discesa, raddoppiando così il tasso di trasferimento dati

ROM (Read-only-memory) non possono essere cancellate né modificate né intenzionalmente né accidentalmente

① i dati sono inseriti durante la sua fabbricazione → ciò avviene esponendo alla luce un materiale fotosensibile attraverso una maschera contenente il PATTERN di bit desiderato, e intendendo quindi la superficie esposta

→ PROM (Programmabile): può essere programmabile sul posto → contengono un ARRAY di piccoli fusibili, che possono essere bruciati una volta fatto selezionando riga e colonna e applicando un alta tensione a un particolare pin del chip

→ EPROM (Erasable) non solo possono essere programmati, ma anche cancellati. Quando si espone la piccola lente al quarzo che si trova nella EPROM ad un'intensa luce ultravioletta per 15 min, tutti i bit assumono valore 1

→ EEPROM (Electric) possono essere cancellate applicando impulsi elettrici. Invece di dover inserire il chip in una camera speciale a luce ULTRA-VIOLETTA → È detta anche MEMORIA FLASH è cancellabile a blocchi e riscrivibile

CHIP CPU: tutte le CPU moderne sono contenute in un unico chip. Ogni chip di CPU ha un insieme di pin attraverso i quali possono tutte le relative comunicazioni verso il mondo esterno → i pin di una CPU possono essere di 3 tipi: indirizzi, dati e controllo; questi pin sono collegati ad qualche pin sulla memoria o in altro posto, mediante un insieme di cavi paralleli chiamato bus.

→ due principali parametri che determinano le prestazioni di una CPU sono il numero di pin d'indirizzo e il numero di PIN di dati. Con un chip con n pin d'indirizzo può indirizzare fino a 2^n locazioni di memoria

→ oltre ai pin d'indirizzo e dei dati, le CPU sono dotate anche di alcuni pin di controllo. Questi regolano il flusso e la temporizzazione dei dati da e verso la CPU. I pin di controllo possono essere raggruppati: ① controllo del bus ② interrupt ③ arbitraggio del bus ④ comunicazione con il coprocessore ⑤ stato

- Inoltre i pin per il controllo del bus mandano sul bus dei segnali per notificare quando la CPU vuole leggere/scrivere in memoria

① INTERRUPT: sono degli input che giungono alla CPU dalle periferiche. Non appena il dispositivo ha completato la propria operazione asserisce un segnale su uno di questi pin in modo da interrompere la CPU e permettergli di utilizzare il dispositivo

② ARBITRAGGIO: dei bus sono necessari per regolare il traffico sul bus → ai fini dell'arbitraggio la CPU conta quanto un altro dispositivo e anch'esso deve fare esplicita richiesta di utilizzo del bus

BUS: è un collegamento elettrico che unisce diversi dispositivi. I bus possono essere classificati in base alla loro funzione. I primi PC avevano un bus esterno chiamato BUS di SISTEMA. I bus moderni hanno invece un bus specifico tra la CPU e la memoria e un altro bus per le periferiche

→ PER i BUS ESTERNI: bisogna definire delle precise regole di funzionamento che devono essere rispettate dai dispositivi a loro collegati → possibile che anche schede progettate da altri produttori possano collegarsi correttamente al bus. → REGOLE → PROTOCOLLO DEL BUS.
Alcune periferiche che si collegano al bus sono attive e possono iniziare un trasferimento dati, mentre altre sono passive e restano in attesa di una richiesta ATTIVE: MASTER / PASSIVE: SLAVE

- visto che i segnali digitali generati dalle periferiche sono troppo deboli per alimentare un bus → per questo molti MASTER sono connessi al bus mediante un chip chiamato driver del bus - (funge da amplificatore digitale) e un altro RICEVITORE DEL BUS.
Inoltre le periferiche che possono svolgere sia il ruolo di master sia quello di slave si utilizza un chip chiamato TRASMETTITORE - RICEVITORE DEL BUS → questi dispositivi possono svolgere sono a tre stati per permettere al bus di essere Given (Sconnesso)

! non è necessario che vi sia uno corrispondenza uno a uno tra i PIN della CPU e i segnali del bus

Le principali decisioni da prendere nella progettazione di un BUS riguardano:

1.- AMPIEZZA DEL BUS: Maggiore è il numero di linee d'indirizzo di un bus, maggiore sarà la quantità di memoria che la CPU potrà indirizzare direttamente. Se un BUS ha n linee d'indirizzo, una CPU può indirizzare 2^n diverse locazioni di memoria

- Occorre trovare un compromesso tra la dimensione massima di memoria e il costo del sistema
→ Esistono due modi per aumentare la larghezza di banda dei dati su un bus: diminuire il periodo di clock del BUS, oppure aumentare la larghezza dei dati del bus
→ DISALLINEAMENTO DEL BUS (BUS SKEW) dovuto alle diverse velocità dei bus quindi l'aumento della velocità del BUS non è una strada percorribile → l'approccio più utilizzato è quello di aggiungere nuove linee di dati del bus
→ BUS MULTIPLEXATO: in questa architettura invece di tenere separate le linee di indirizzo e quelle dei dati, si utilizza un certo numero di linee per entrambi. L'utilizzo del multiplexing riduce l'ampiezza del bus ma rende il sistema più lento

2.- TEMPORIZZAZIONE DEL BUS-
I bus possono essere separati in due categorie distinte in base alla loro temporizzazione

a) BUS SINCRONO: ha una linea pilotata da un oscillatore a cristalli: su questa linea un segnale consiste in un onda quadra con frequenza generalmente compresa tra 5 e 100 MHz

- tutte le operazioni sul bus richiedono un numero intero di questi cicli → CICLI DI BUS

Calcolatori Elettronici — ARCHITETTURA DEI CALCOLATORI — CAP 3 — 10/

2. - TEMPORIZZAZIONE DEL BUS -

(a) BUS PCI molto diff e funziona a 33MHz / 66MHz

ESEMPIO: Lettura mem. = 15 nS a partire dal momento in cui è indirizzo è stabile. A partire da T_2 la CPU fornisce l'indirizzo della parola sulle linee d'indirizzo (visualizz. con gruppi di 1's) Dopo che le linee di indirizzo si sono stabilizz. sui bus valori vengono asserite $\overline{MREQ}$ e $\overline{RD}$. Nel momento in cui la mem. è occupata asserisce $\overline{WAIT}$ → quest'operazione inserisce alcuni stati di attesa finché la memoria non completa l'operazione e neghi $\overline{WAIT}$. Successivamente la mem. mette i dati su linee apposite. Nel fronte di discesa la CPU legge (ovvero nega $\overline{MREQ}$ e $\overline{RD}$)

(b) BUS ASINCRONI: non ha un orologio principale; i cicli di bus possono avere una qualsiasi lunghezza.

(!) MIGLIORE RISPETTO AL BUS SINCRONO: nel momento in cui un bus ha un insieme eterogeneo di dispositivi, alcuni veloci, altri più lenti, il bus deve essere regolato alla velocità della periferica più lenta, e le altre pur essendo più veloci non possono sfruttare le loro potenzialità

→ Il master bus invece di leggere ogni operazione al clock, dopo aver asserito l'indirizzo, $\overline{MREQ}$, $\overline{RD}$ e tutto ciò che gli necessita invia una speciale segnale $\overline{MSYN}$ (Master Synch) Quando lo slave la vede esegue il lavoro richiesto alla massima velocità e una volta terminato il proprio compito asserisce $\overline{SSYN}$ (Slave Synchronization)

- Quando il master vede che $\overline{SSYN}$ è stato asserito, sa che i dati sono disponibili e può quindi memorizz. Successivamente nega la linea d'indirizzo $\overline{MREQ}$, $\overline{RD}$ e $\overline{MSYN}$. Quando lo slave vede la negazione di $\overline{MSYN}$ sa che il ciclo è stato completato e quindi nega $\overline{SSYN}$

(!) nei diagr. asincr. i segni freccia per indicare cause ed effetti → FULL HANDSHAKE → non dip. dalla temporizzazione. Ogni evento è causato da un evento precedente e non da un impulso di CLOCK

(3) ARBITRAGGIO DEL BUS: interviene quando due o più dispositivi vogliono diventare master del BUS nello stesso momento. Questo meccanismo può essere:

→ CENTRALIZZATO: c'è un arbitro non ha alcun modo di sapere quanti dispositivi hanno richiesto il bus. Utilizza COLLEGAMENTO A FESTONE, ha la proprietà di assegnare ai dispositivi diversi priorità in base alla loro vicinanza all'arbitro (→ più vicino: + alta)

→ ~~DECENTRALIZZATO~~: utilizza per assegnare il bus che le priorità sono implicitamente determinate dalla distanza rispetto dall'arbitro, definiscono LIVELLI DI PRIORITÀ

→ Es: alcuni arbitri hanno una terza linea che viene asserita da un dispositivo nel momento in cui ottiene una connessione e si impadronisce del bus. Subito dopo aver asserito questa linea di conferma. Il dispositivo può negare la linea di richiesta e cosi il richiesto è che altri dispositivi possono richiedere il bus mentre il primo lo sta ancora aspettando. Nel momento in cui termina il trasf. in corso il successivo MASTER del bus sarà già selezionato

- Poiché la CPU ha priorità più bassa in modo che possa usare il bus soltanto quando nessun altro lo vuole → poiché è molto più veloce

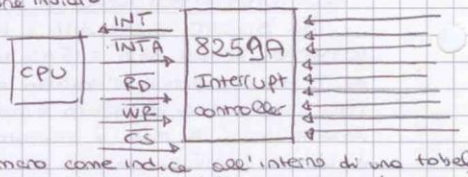
→ DECENTRALIZZATO: con un numero di linee abbastanza ridotto si può generare una gerarchia di priorità → quando un dispositivo vuole ottenere il bus asserisce la linea di richiesta → non c'è più arbitro

- Il numero di dispositivi non può superare il n° delle linee di richiesta
- Quando un dispositivo vuole il BUS: ① invia una richiesta di bus ② verifica che il bus è libero. Se IN non è asserito non diventa master e nega OUT. Se IN è asserito diventa master, nega OUT e asserisce BUSY → meccanismo più semplice e veloce

→ BLOCK TRANSFERS: utilizzato per trasferire blocchi di dati. Il numero di parole viene specificato durante T_1. Dopo la prima viene trasferita una parola ogni ciclo (invece di una ogni 3 cicli)
Per leggere quattro parole occorrono 6 cicli invece di 8.
Il segnale di BLOCK viene asserito per chiedere un block transfer

OPERAZIONI DEL BUS

GESTIONE DELLE INTERRUZIONI: Chip controllore di interruzioni gestisce 8 linee di interrupt ① INT: interruzione inviato alla CPU
② INTA: acknowledge della CPU
③ Numero del dispositivo sul bus ④ Il numero viene usato come indice di un vettore di interrupt
⑤ IR0 - IR7: linee di interrupt



L'Hardware della CPU inserita quindi quel numero come indice all'interno di una tabella di puntatori chiamata vettore di interrupt per cercare l'indirizzo della procedura da eseguire per poter gestire l'interrupt → i registri all'interno del chip sono scrivibili dalla CPU per programmare il controllore

CPU

PENTIUM IV: fu introdotto nel 2000 sottoforma di CPU a 1,5 GHz con 42 milioni di transistors con larghezza di linea da 0,18 micron
↳ corrisponde alla grandezza dei collegamenti tra i transistors
↳ adotto una nuova microarchitettura chiamata NETBURST
↳ dotato di una pipeline più profonda e due ALU
↳ supporta HYPERTHREADING → fornisca due insiemi di registri e alcune altre risorse interne per permettere di eseguire l'esecuzione di due programmi molto velocemente

SNOOPING: Operazione di "spionaggio" per garantire la coerenza della memoria in una multiprocessore → spionaggio sul bus della memoria per cercare eventuali riferimenti alla parola che ha memorizzato nella propria cache
↳ architettura CISC a 32 bit / bus memoria 64 bit
↳ ottimizzata FLOATING POINT IEEE 754

PIPELINE SUL BUS MEMORIA: utilizzate per evitare che la CPU rimanga in attesa a causa dei dati della mancanza dei dati
RICHIESTE DI MEMORIA = TRANSAZIONI composte da STADI: ① fase di arbitraggio del bus ② fase di richiesta ③ fase di segnalazione dell'errore ④ fase di investigazione ⑤ fase di risposta ⑥ fase dei dati → non sempre si usano tutte le fasi

SUN - ULTRASPARC III: Architettura RISC a 64bit / superscalare e pipelined
TARGET: sistemi multiprocessore a memoria condivisa → fino a 29 milioni di transistor
indirizzi da 43 bit → fino a 8 TB di memoria
- aggiunto di istruzioni VIS 2.0 per le applicazioni grafiche 3D, la decodifica MPEG in tempo reale, la compressione dei dati etc
- concepito per essere impiegato nei server multiprocessore dotati di una grande memoria condivisa
- è in grado di eseguire in continuazione 4 istr per ciclo di clock
- possiede 6 pipeline interne due comprendono 2 pipeline a 14 stadi per operazioni su interi due per le operazioni in virgola mobile, una per LOAD/STORE

Per accedere alla memoria la CPU deve inizialmente utilizzare i pin per l'arbitraggio del bus in modo da eseguire il diritto di a proseguire → specifica il tipo di richiesta e assenza i pin per la validità d'indirizzo. ~~Questi~~ Questi PIN sono bidirezionali dato che in un sistema multiprocessore altre CPU devono poter accedere alle CACHE REMOTE

8051 è il microcontrollore più diffuso → grazie al suo basso costo → ideale EMBEDDED
ha 40 pin → 16 linee di indirizzo → fino a 64 KB di memoria. Il bus dei dati è largo 8 bit e quindi i trasferimenti dati tra CPU e memoria avvengono un byte alla volta

Calcolatori elettronici

ARCHITETTURA DEI CALCOLATORI
CAP 3 11/

8051 → presenta di 32 linee I/O organizzate in gruppi di 8 bit ciascuno
A ognuno di ta queste linee e possibile collegare dispositivi che possono fornire un input o un output → (linee bidirezionali)
Il primo segnale A contiene 16 linee d'indirizzo per indirizzare i byte della memoria esterna che si intende leggere o scrivere
Le 8 linee D sono invece utilizzate per il trasporto dei dati
Le 8 linee d'indirizzo meno significative sono multiplexate negli stessi pin dei dati

- Il segnale ALE (Address Latch Enable) è utilizzato in presenza di una memoria esterna → viene asserito dalla CPU per indicare che l'indirizzo è valido → le memorie esterne lo utilizzano per stabilizzare le linee degli indirizzi

BUS

BUS ISA: (Industry Standard Architecture) derivato dal bus PC/AT. Operando volto a mantenere la compatibilità tra le macchine e le schede esistenti
bus sincrono → 62+36 linee: 16 dati e 24 indirizzi
Successivamente il bus venne esteso a 32 bit con l'aggiunta di alcune nuove funzionalità prese il nome di EISA (Extended ISA)

BUS PCI: Nel 1990 la Intel progettò un nuovo bus per app video con una larghezza di banda molto più ampia rispetto all'EISA e lo chiamò PCI (Peripheral Component Interconnect). Questo bus può funzionare fino a 66MHz e può gestire trasferimenti di 64bit per una larghezza di banda di 528 MB/s
↳ BRIDGE: Il bridge PCI connette la CPU la memoria e il bus PCI, mentre il bridge ISA connette il bus PCI al bus ISA e supporta uno o due dischi IDE
↳ Supporta due diverse tensioni 5 - 3,3 volt → intercompatibilità verso privato
↳ È sincrono tutte le transazioni sul bus PCI avvengono tra un master detto initiatore e uno slave → destinatario. P.

- Per ridurre il numero di ~~linee~~ pin, le ~~due~~ linee dati-indirizzi sono multiplexate

FUNZIONAMENTO: Nel ciclo 1 di un operazione di lettura il master immette l'indirizzo sul bus. Nel ciclo 2 il master rimuove l'indirizzo e il bus è invertito in modo che lo slave lo possa ricevere; infine nel ciclo 3 lo slave genera in OUTPUT i dati richiesti

BUS AGP: (Accelerated Graphic Port) bus dedicato per le schede grafiche
Prevede la presenza di un nuovo slot con un connettore diverso da quello PCI sulla scheda madre
→ A differenza del PCI adotta una tecnica pipeline
→ modalità fino a 8x (AGP 3.0) 2,1 GB/sec
→ le schede AGP adottano il sistema DIRECT MEMORY EXECUTE (DIME) che consente al controller di svolgere le elaborazioni usando la memoria principale.
→ Il protocollo prevede la presenza di un solo dispositivo

In questa architettura diventa centrale il chip bridge che collega i 5 slot principali del sistema al suo interno il chip bridge è diviso in 2 parti quella verso la memoria e quella verso l'IO

① Il bus PCI è sincrono e tutte le transazioni sul Bus avvengono tra un master detto iniziatore e uno SLAVE chiamato destinatario.
- Inoltre per ridurre il numero totale di PIN, le linee degli indirizzi e dei dati sono MULTIPLEXATE

ARBITRAGGIO BUS PCI: Ogni dispositivo PCI ha due linee dedicate che lo collegano all'arbitro una REQ# per richiedere il bus e l'altra GNT# per ricevere la concessione. Inoltre le specifiche PCI non definiscono l'algoritmo che l'arbitro deve utilizzare → il bus viene concesso per una transazione alla volta. I SEGNALI FONDAM
- → CLK (clock) le transazioni iniziano in corrispondenza del fronte di discesa del CLK
- → AD (32 segnali) dedicati agli indirizzi e ai dati → (TEMPO di TURN AROUND)
- → C/BE# utilizzato per due scopi distinti nel primo ciclo LEGGI/ABILIT nel secondo SCRIVI/ABILITA
- → FRAME# asserito dal master per iniziare una transazione e il bus comunica allo slave che il master è pronto che è indirizzo e i comandi del bus sono validi
- → IRDY# in scrittura asserito quando i dati si trovano sul bus, in lettura asserito con FRAME
- → DEVSEL# annuncia che lo slave ha riavuto il proprio indirizzo sulle linee AD ed è pronto a iniziare una transazione
- → TRDY# viene asserito nella lettura per annunciare che i dati sono disponibili sulle linee AD mentre nella scrittura per annunciare che è pronto a accettare i dati

Quando durante il periodo T_1 si verifica il fronte di discesa del clock, il master inserisce l'indirizzo di memoria su AD e il comando del BUS su C/BE#, asserisce inoltre FRAME # per iniziare la transazione sul bus. Durante T_2 il master rilascia l'indirizzo del bus per permettere che venga invertita in modo che lo SLAVE possa prelevarlo durante T_3. Durante T_3 lo slave asserisce DEVSEL# in modo che il master sappia che ha riavuto il proprio indirizzo e che si sta adoperando per rispondere. Inoltre scrive i dati sulle linee AD e asserisce TRDY # per notificare il fatto al master. Se lo slave non è in grado di rispondere velocemente deve tuttavia asserire DEVSEL# per annunciare la propria presenza ma deve lasciare TRDY# negato finché non possa fornire i dati richiesti

BUS PCI EXPRESS: Il cuore del BUS PCI EXPRESS sta nella rinuncia al bus PARALLELO con i suoi numerosi master e slave → novità rappresentata da un commutatore connesso al bridge. Ogni chip I/O ha una connessione dedicata punto-punto con il commutatore che consiste in una coppia di canali unidirezionali

→ DIFF. CONTRA PCI/PCI EXPRESS: ① commutatore centralizzato al posto di un bus con più connessioni e l'uso di strette connessioni seriali punto-a-punto al posto di un unico bus parallelo. ② il modello del PCI EXPR. → dispositivo che spedisce pacchetti, dove l'intestazione contiene le informazioni di controllo

Un **PROTOCOLLO** è un insieme di regole che governano la comunicazione tra due parti. Uno stack di protocolli è una gerarchia di protocolli che gestisce diversi problemi in livelli distinti
- LIVELLO FISICO (inferiore): tratta lo spostamento dei bit da un mittente a un destinatario lungo una connessione punto a punto.
  - → CODIFICA 8b/10b in cui si utilizza un simbolo a 10 bit per codificare ed evitare problemi riguardanti l'invio di tutti 1 o 0
- LIVELLO DI TRASMISSIONE: si occupa della trasmissione dei pacchetti → assegna inoltre ai pacchetti un codice a correzione di errore CRC (Cyclic Redundancy Check). Se il pacchetto inviato e quello ricevuto corrispondono restituisce un pacchetto di ACKNOWLEDGMENT per confermare la corretta ricezione del pacchetto
- LIVELLO DI TRANSAZIONE: gestisce le azioni sul bus. La lettura di una parola dalla memoria richiede due transazioni: una iniziata dalla CPU o dal canale DMA per richiedere alcuni dati e una iniziata dal mittente per fornire i dati
  - → Per iniziare la trasmissione può dividere ciascuna corsia in più circuiti VIRTUALI fino ad un massimo di 8 ciascuno dei quali gestisce una diversa classe di traffico
- LIVELLO SOFTWARE: che interfaccia il sistema PCI Express al sistema operativo può emulare il bus PCI in modo che sia possibile utilizzare sistemi operativi non modif

① Una differenza sostanziale tra una rete e PCI EXPRESS è che nel campo delle reti il codice dei vari livelli è quasi sempre una parte del SOFTWARE del sistema operativo, mentre con PCI EXPRESS fa interamente parte dell'hardware del dispositivo

BUS USB: (Universal Serial BUS), noto come esigenza primaria di abbassamento dei costi per connettere periferiche a bassa velocità.

BUS USB: standard introdotto nel 1998, progettato per dispositivi a bassa velocità come tastiere mouse etc. Un sistema USB è composto da un HUB principale che si collega al bus del sistema e che possiede delle prese per i cui che connettono i dispositivi I/O o HUB per espansione.

- Il cavo consiste in 4 collegamenti: due per i dati uno per l'alimentazione e uno per la terra
- Quando viene collegato un nuovo dispositivo l'hub principale nota l'evento e interrompe il sistema operativo; quest'ultimo interroga il dispositivo per sapere di che periferica si tratta e di quanta banda ha bisogno.
- I dati vengono inviati in FRAME, un frame è associato a un contatto di bit e consiste in pacchetti il primo dei quali viaggia dall'HUB principale verso il dispositivo

SOF | IN | DATA | ACK

→ USB supporta 4 tipi di frame:

1. FRAME di CONTROLLO: usati per configurare i dispositivi, assegnare loro dei comandi e interrogarli sul loro stato
2. FRAME ISOCRONI: servono per dispositivi funzionanti in tempo reale che devono spedire o accettare dati a precisi intervalli temporali
3. FRAME BULK sono utilizzati per trasferimenti di grandi dimensioni per dispositivi che non richiedono un funzionamento in tempo reale
3. FRAME INTERRUPT: sono necessari in quanto USB non supporta gli interrupt

Un frame è composto da uno o più pacchetti, alcuni dei quali possono spostarsi in entrambe le direzioni

- VARI TIPI DI USB: 1) 1,5 basso velocità 2) 2x/480 mbit/s 3) 3x / 5 Gbit/s

INTERFACCE

→ **CHIP di I/O**: è attraverso questi chip che il calcolatore comunica con il mondo esterno

UART (Universal Asynchronous Receiver Transmitter) è un chip che può leggere un byte da un Bus di dati e generarlo in output, un bit alla volta

USART: (Universal Synchronous Asynchronous Receiver transmitters), possono gestire trasmissioni sincrone utilizzando vari protocolli oltre a supportare tutte le funzionalità dei CHIP

PIO: (Parallel Input/output) ex Intel 8255A → il modo più semplice di utilizzare questo chip è con 3 porte indipendenti a 8 bit A,B,C, ciascuna delle quali è associata un registro latch a 8 bit. Per impostare le linee su una porta, la CPU scrive semplicemente un numero a 8 bit nel registro corrispondente a tale numero rimane sulle linee di output finché il registro non viene riscritto

→ Il chip PIO può essere selezionato come un vero dispositivo I/O o come parte della memoria.

La selezione del CHIP è assente sulla memoria e sui CHIP di I/O.
Se per PIO scegliamo l'approccio dell'I/O mappato in memoria dobbiamo assegnare 4 byte della memoria necessari per indirizzare tutte le porte di e il registro di controllo.
Ci sono due tip. di codifiche ① Decodifica totale utilizza molte linee ② DECODIFICA PARZIALE → utilizza pochissime linee in quanto si possono aggiungere a posteriori altri dispositivi in quanto cambia l'assegnazione del CHIP SELECT.

CAP 4

LIVELLO DI MICROARCHITETTURA

- a livello della microarchitettura studiamo come la CPU "implementa" le istruzioni macchina mediante i dispositivi digitali "HARDWARE" a sua disposizione

MICROARCHITETTURA DELLA CPU → composta da alcuni registri, una ALU, dei bus interni e alcune linee di controllo. Le istruzioni macchina comandano il funzionamento della CPU e il percorso dei dati

MICROPROGRAMMAZIONE: in un'architettura microprogrammata le istr macchina non sono eseguite direttamente dall'HW → ma solo quando sono microistruzioni
- dall'esecuzione di ciascuna istruzione macchina corrisponde l'esecuzione di diverse microistruzioni

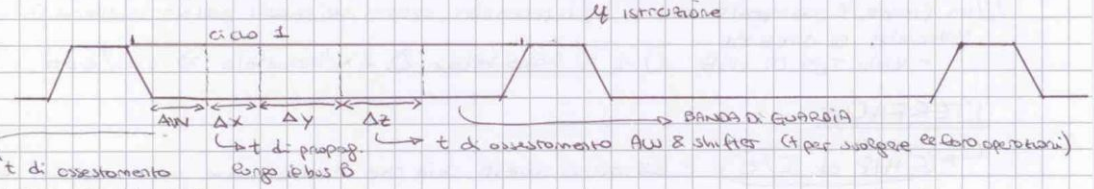
VANTAGGI: flessibilità, possibilità di gestire istruz. macchina complesse
SVANTAGGI: esce esecuzione relativamente lenta; ciascuna istruzione richiede più passi elementari

ex:
↳ μ-ARCHITETTURA: Implementazione di un JVM con sole istruzioni su interi
- REGISTRI:
  1) PC (program Counter) contiene l'indirizzo di memoria della prossima istruzione
  2) MBR (memory byte register) sarebbe l'IR (istruction register) contiene l'indirizzo da eseguire
  3) MAR (memory address register) contiene l'indirizzo dell'istruzione (della memoria) che deve essere acceduto
  4) MDR (memory data register) contiene il dato che deve essere acceduto / memorizzato
  5) H (loading registers / accumulator) unico che presenta le sue contenuto sia in uscita che in ingresso all'ALU

  Questo approccio va bene per architetture semplici; grazie ad esso il n° di bus interni diminuisce per fare in modo che ogni registro utilizzi 2 BUS uno x input dell'ALU

SEGNALI DI CONTROLLO:
- ABILITATORI: operano determinate parti per consentire il passaggio dei dati sui bus
  OSS: se 1 può essere asserito contemporaneamente
- ABILITATORI: operano dati forse per consentire la scrittura del dato in OUT dell'ALU sui registri
  OSS: può essere asserito anche più di uno contemporaneamente

TEMPORIZZAZIONE DEL CICLO BASE: tutte le operazioni di un ciclo devono avvenire in un ciclo di clock o in suoi multipli. In ciascun ciclo viene eseguita una μ istruzione



STRUTTURA DELLE μ-ISTRUZIONI: una micro istruzione contiene tutti i segnali di controllo da inviare al data path durante il ciclo. → le informazioni per la scelta della μ-istruzione successiva

BITS (9) (3) (8) (9) (3) (4)

next ADDRESS	JMPC	JAMN	JAMZ	SLL8	SRA1	F_0	F_1	ENA	ENB	INVA	INC	H	OPC	TOS	CPP	LV	SP	RP	MDR	MAR	WRITE	READ	FETCH	B bus

Addr (indirizzo prox istruz) | JAM (scelta 3x istr) | ALU (comandi ALU & shifter) | C (registri da caricare) | Mem (controllo della mem) | B (registro da inviare su B)

Il meccanismo mostrato in figura è quello che permette di trasformare un'istruzione in una microistruzione
- si mette l'istruzione nel MPC (micro-program-counter) registro che scelgie una locazione di una memoria interna chiamata CONTROL STORE una memoria ROM (in questo caso $2^9 = 512$ celle ciascuna con una parola di 30 bit dim (MPC) = 9 bit → memoria ha dimensione di 29
- ADDR: carica le nuove istruzioni, quando non esistono più → vengono caricate le altre
- J: indica che a meno di imprevisti la prossima istruzione da eseguire è nell'Addr se il risultato è Not → porterebbe ad un'altra istruzione

OSS - l'ordine di esecuzione delle micro istruzione non è esattamente sequenziale

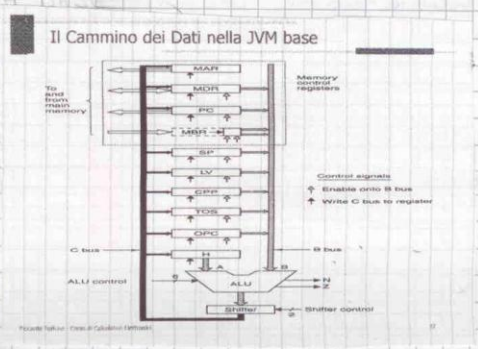
Come faccio a migliorare le prestazioni? ↑ velocità? non solo ma max $\frac{\text{velocità}}{\text{prezzo}}$

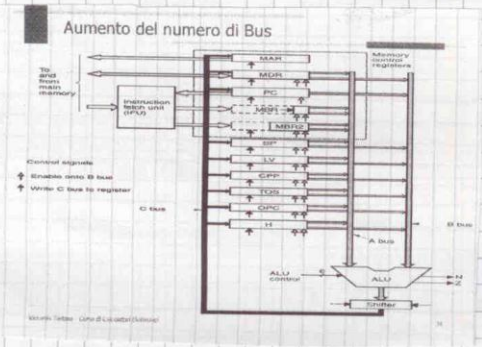
∃ tre tecniche:
1) Introdurre componenti dedicate che svolgono determinate operazioni (↑ costi: dee data path)
2) ↑ la frequenza (ormai ha poco importanza)
3) Introduzione parallelismo (pipeline / duplicazione componenti) → più importante

Calcolatori Elettronici

ARCHITETTURA DEI CALCOLATORI

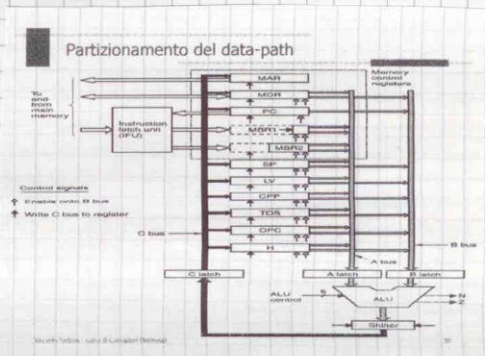
CAPO 13/

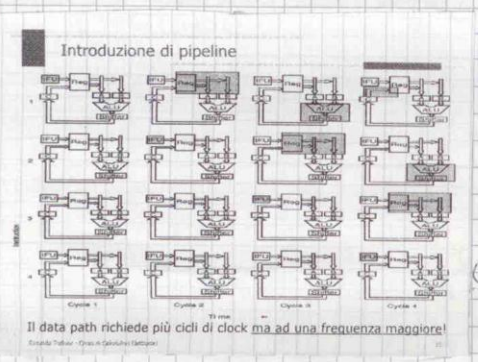




ISTRUCTION FETCH UNIT

- Il carico della ALU può essere alleviato introducendo una unità indipendente che carica le istruzioni da eseguire
- Una possibile IFU incrementa autonomamente il PC e anticipa il caricamento delle istruzioni





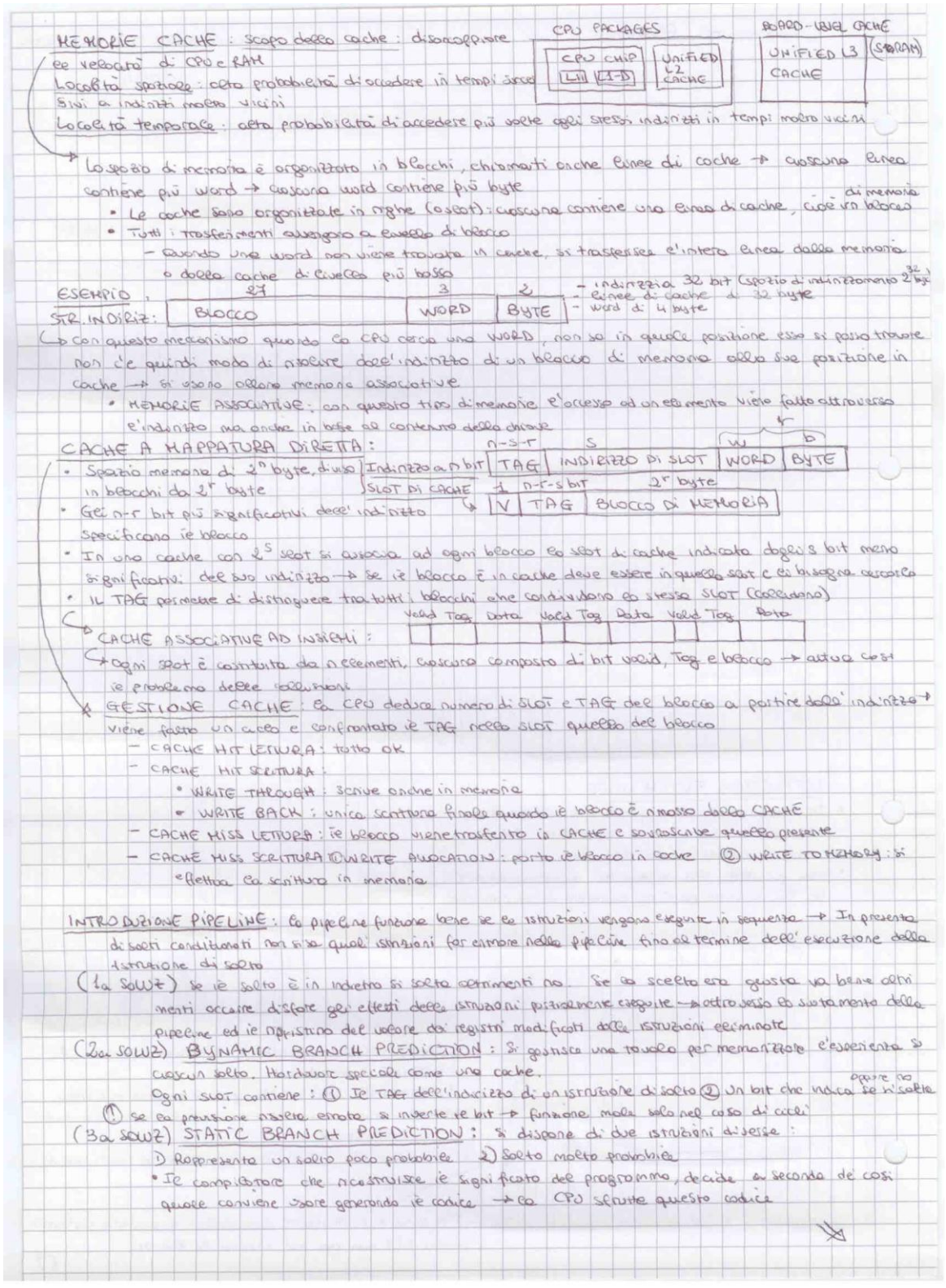
Per aumentare la capacità di elaborazione della CPU

- PIPELINE a molti stadi
- ARCHITETTURE SUPERSCALARI: più di pipeline parallele

LATENZA: tempo necessario a completare l'elaborazione di un istruzione

BANDA DELLA CPU: n° di istruzioni elaborate nell'unità di tempo

Pipeline e architetture superscalari aumentano la banda della CPU ma non aiutano la potenza


MEMORIE CACHE : scopo della cache : disaccoppiare
velocità di CPU e RAM
Località spaziale : alta probabilità di accedere in tempi brevi a indirizzi molto vicini
Località temporale : alta probabilità di accedere più volte agli stessi indirizzi in tempi molto vicini
CPU PACKAGES
CPU CHIP
L1I
L1D
UNIFIED L2 CACHE
BOARD-LEVEL CACHE
UNIFIED L3 CACHE
(SRAM)
Lo spazio di memoria è organizzato in blocchi, chiamati anche linee di cache → ciascuna linea contiene più word → ciascuna word contiene più byte
Le cache sono organizzate in righe (o set): ciascuna contiene una linea di cache, cioè un blocco di memoria
Tutti i trasferimenti avvengono a livello di blocco
Quando una word non viene trovata in cache, si trasferisce l'intera linea dalla memoria o dalla cache di livello più basso
ESEMPIO
STR. INDIRIZ:
BLOCCO
WORD
BYTE
- indirizza 32 bit (spazio di indirizzamento 2^32 byte)
- linee di cache di 32 byte
- word di 4 byte
con questo meccanismo quando la CPU cerca una WORD, non so in quale posizione esso si possa trovare non c'è quindi modo di ottenere dall'indirizzo di un blocco di memoria alla sua posizione in cache → si usano allora memorie associative
MEMORIE ASSOCIATIVE: con questo tipo di memoria l'accesso ad un elemento viene fatto attraverso l'indirizzo ma anche in base al contenuto della cache
CACHE A MAPPATURA DIRETTA :
n-s-r
s
w
b
Spazio memoria di 2^n byte, diviso in blocchi da 2^r byte
Indirizzo a n bit
TAG
INDIRIZZO DI SLOT
WORD
BYTE
SLOT DI CACHE
n-r-s bit
2^r byte
V
TAG
BLOCCO DI MEMORIA
Ci sono n-r bit più significativi dell'indirizzo
Specificano il blocco
In una cache con 2^s slot si associa ad ogni blocco lo slot di cache indicato dagli s bit meno significativi del suo indirizzo → se il blocco è in cache deve essere in quello slot e di bisogna cercarlo
Il TAG permette di distinguere tra tutti i blocchi che condividono lo stesso SLOT (collisioni)
Valid Tag Data Valid Tag Data Valid Tag Data
CACHE ASSOCIATIVE AD INSIEMI :
Ogni slot è costituito da n elementi, ciascuno composto di bit valid, Tag e blocco → attua così le politiche delle sostituzioni
GESTIONE CACHE : la CPU deduce numero di SLOT e TAG del blocco a partire dall'indirizzo → viene fatto un accesso e confrontato il TAG nello SLOT quello del blocco
CACHE HIT LETTURA : tutto OK
CACHE HIT SCRITTURA :
WRITE THROUGH : scrive anche in memoria
WRITE BACK : unica scrittura finale quando il blocco è rimosso dalla CACHE
CACHE MISS LETTURA : il blocco viene trasferito in CACHE e sovrascrive quello presente
CACHE MISS SCRITTURA (WRITE ALLOCATION) : porto il blocco in cache ② WRITE TO MEMORY : si effettua la scrittura in memoria
INTRODUZIONE PIPELINE : la pipeline funziona bene se le istruzioni vengono eseguite in sequenza → in presenza di salti condizionati non si sa quali istruzioni far entrare nella pipeline fino al termine dell'esecuzione della istruzione di salto
(1a SOLUZ) se il salto è in indietro si salta sempre no. Se la scelta era giusta va bene altrimenti occorre disfare gli effetti delle istruzioni prematuramente eseguite → altro peso in svuotamento della pipeline ed il ripristino dei valore dei registri modificati dalle istruzioni eseguite
(2a SOLUZ) DYNAMIC BRANCH PREDICTION : si gestisce una tavola per memorizzare l'esperienza di ciascun salto. Hardware speciale come una cache.
Ogni SLOT contiene : ① la TAG dell'indirizzo di un istruzione di salto ② un bit che indica se si salta oppure no
① se la previsione risulta errata si inverte il bit → funziona molto solo nel caso di cicli
(3a SOLUZ) STATIC BRANCH PREDICTION : si dispone di due istruzioni di salto :
1) Rappresenta un salto poco probabile 2) Salto molto probabile
Il compilatore che ricostruisce il significato del programma, decide a seconda dei casi quale conviene usare generando il codice → la CPU esegue questo codice

Calcolatori Elettronici — ARCHITETTURA DEI CALCOLATORI

CAP 4 — 14/

IN-ORDER EXECUTION: in architetture con PIPELINE una prima scelta è di fare le istruzioni siano eseguite e completate nello stesso ordine in cui sono state decodificate. IN-ORDER-ISSUE / IN ORDER RETIRE

- Scelta molto limitante sotto il profilo delle prestazioni
- Si deve rimandare l'inizio dell'istruzione se si verifica o presenta di un vincolo con istruzioni precedenti non ancora completate

VINCOLI TRA ISTRUZIONI:

(RAW): uno dei registri sorgente viene scritto da un'altra istruzione (Read After Write)
(WAR): il registro destinazione viene letto da un'altra istruzione (Write After Read)
(WAW): il registro destinazione viene scritto da un'altra istruzione (Write after Write)

OUT-OF-ORDER EXECUTION: Ricordando i vincoli di ordine si hanno migliori prestazioni, si deve garantire sempre una esecuzione corretta
→ I vincoli RAW, WAR, WAW devono continuare ad essere rispettati
→ Si deve tener conto della sottrattura fatta da istruzioni in attesa
① Register Renaming: si risolvono conflitti WAR e WAW sostituendo con registri SCRATCH (segreti) mentre per le RAW posso solamente mettere in attesa

SPECULATIVE EXECUTION: I programmi possono essere scomposti in basic blocks puramente sequenziali → Il programma è un grafo orientato i cui nodi sono basic-blocks
- L'esecuzione OUT-OF-ORDER funziona bene all'interno dei basic block
- Principio: eseguire anche istruzioni che forse non servono

CPU PENTIUM IV: anche se è obsoleto tuttavia è la base di tutte le nuove architetture nuove
→ Macchina CISC a 32 bit con un bus in predisposto a 64 bit
→ Set di istruzioni molto disordinato in quanto basato vedere da Borghetto Da quale risulta essere molto vandalica
→ 8 registri ufficiali + molti altri segreti
Ha un nucleo RISC tuttavia bisogno stare attenti poiché la tecnica adottata è IBRIDA (CISC)

4 sottosistemi principali: ① Interfacciamento con la memoria e presenta una cache L2 ② nella seconda ho un'astrazione sotto composizione blocco FETCH/DECODE → una Trace cache, una micro ROM che serve per scindere le istruzioni macro istruzioni, ed un schema unità che si occupa della predizione di salto → in base a questo viene fatto il FETCH ③ si occupa dell'esecuzione fuori ordine, in cui sono presenti dei schedulatori → si decide l'ordine in cui vengono eseguite
④ si occupa dell'esecuzione dove ci sono le ALU ed una CACHE di primo livello ed una volta che le istruzioni sono state trovate vengono ritirate
① GENERAZIONE IN-ORDINE, AVVIO FUORI-ORDINE, RITIRO IN ORDINE

CPU ULTRASPARC III CU: Macchina RISC a 64bit → bus a 128 bit
→ le istruzioni sono tutte fisse 4 bytes → 32 bit, con un HW per istr. multimediali
Si ha una decomposizione: ① CACHE di secondo livello nella quale le istr. vengono caricate e c'è anche un interfaccia ② CACHE di 1° livello in cui vengono sottodecomposte ③ Predizione con HW dedicato ④ le istruzioni vengono messe in un BUFFER
① le istruzioni aritmetiche prelevano dati solo da registri se si vuole prendere i dati dalla memoria principale deve utilizzare le LOAD e le STORE unit

CPU 8051: cpu → microprocessore economico → si nasce dalle caratteristiche complesse quindi è un architettura RISC → con 64K transistor —
- 8/16 bit al massimo e pochi registri eterogenei
- unico ciclo di esecuzione diviso in 6 stadi

Organizzazione semplice, non c'è una PIPELINE, ne una CACHE per risparmiare. Le esecuzioni e i fetch avvengono in ordine

MICROARCHITETTURA 8051: i registri sono collegati al BUS principale:

REGISTRI: ACC accumulatore
B: registro di lavoro a 8 bit
SP: punta alla cima dello stack
IR: registro istruzioni
PSW: registro di stato

- PC: program counter autoincrementa con registro increment. autonomo
- BUFFER: registro di lavoro di 16 bit
- 4 PORTE I/O a 8 bit → controllano fino a 32 disp. ext

ESECUZIONE ISTRUZIONE 8051: Esecuzione in un ciclo di clock

1) Nuova istruzione prelevata dalla ROM e inviata nell'IR
2) Decodifica istruzione e incremento PC
3) Preparazione degli operandi
4) Dati in un registro e nell'accumulatore inviati in TMP1 e TMP2
5) ALU esegue l'operazione
6) L'output della ALU viene registrato in un registro e ROM ADDR viene caricato

MICROARCHITETTURA 8051

CAP 5

IL LIVELLO DELLE ISTRUZIONI MACCHINA

ISA (Instruction set architecture): il livello ISA è l'interfaccia tra HW e SW ed inoltre è il livello più basso a cui il processore è "programmabile"

- CRITERI di SCELTA: ① semplicità di implementazione ② efficienza della microarchitettura ③ semplicità di generazione del codice ④ Compatibilità con il passato

definizione: costituisce il riferimento per coloro che scrivono i compilatori (oppure in assembler)

Concetti FONDAMENTALI: Memoria → organizzazione
Registri → quali registri sono visibili al livello ISA e quali funzioni hanno
Indirizzamento → come le istruzioni fanno riferimento ai loro operandi
Istruzioni → repertorio delle istruzioni interpretate dal livello ISA

Modello di esecuzione tipicamente user mode e kernel mode

LINGUAGGIO ASSEMBLATIVO = linguaggio simbolico / LINGUAGGIO MACCHINA = linguaggio in binario

MODELLO DELLA MEMORIA: locazioni di memoria ad cui granularità è di un BYTE

REGISTRI: conoscono solamente i registri ufficiali. All'interno di quest'ultimi esistono REGISTRI di general purpose → servono per risultati intermedi e per dati di uso molto frequente.

special purpose PC: program counter; MDR: Memory Data Register
→ Registro di stato: ci informano dello stato del microprocessore (PSW SW)
→ Registro di Stack o pila: → da cui dipende la struttura a STACK FRAME

STRUTTURA A STACK FRAME: viene inoltrata per ciascuna invocazione viene allocato in una porzione della memoria organizzata a pila.

Lo stack frame contiene: ① I parametri in entrata ed in uscita ② le variabili locali ③ l'indirizzo di ritorno ④ un puntatore allo stack frame del chiamante.

- Lo stack pointer SP punta all'elemento affiorante → } registri special purpose
- Frame Pointer FP punta alla base del frame in cima alla pila)

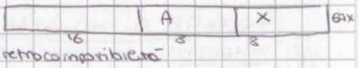
LIVELLO ISA DEL PENTIUM 4: il livello ISA è fortemente influenzato dal vincolo di compatibilità all'indietro. Architettura IA-32
↳ diverse modalità di funzionamento
- Real mode in cui si comporta come il 8088
- Virtual mode 8086: come il 8086 ma intercetta tutte le operazioni dedicate
- Protected mode: si comporta come un P4

Calcolatori Elettronici — ARCHITETTURA DEI CALCOLATORI

LIVELLO DELLE ISTRUZIONI MACCHINA — CAP 5 — 15/

LIVELLO ISA DEL P4: ha 4 livelli di privilegio Kernel = 0, User = 3

- Spazio di memoria di 16 k segmenti di 4GB
- La memoria può essere divisa in segmenti → insieme molto grosso di cartelle di memoria
  - REGISTRI: EAX, EBX, ECX, EDX; EFLAGS
    si possono chiamare in maniera diversa parlando dello stesso indirizzo → EX: se dico AX corrispondente per retrocompatibilità

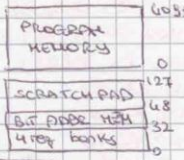


LIVELLO ISA DELL' ULTRASPARC: deriva dalla architettura RISC ed è definita in vari documenti con un architettura 32 a 64 bit

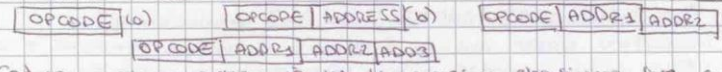
- Esistono soltanto due istruzioni per l'accesso alla memoria LOAD carica dal dalla memoria a registro e store da registro a memoria; tutti gli altri vengono fatte sui registri
- Grosso disponibilità di registri con bassa probabilità di avere delle collisioni
  - REGISTRI: organizzazione dei registri pensata per agevolare le chiamate di funzione.
    In ciascun istante sono visibili attraverso una finestra che varia in occasione delle chiamate di procedura
    → OVERLAPPING REGISTER WINDOWS: la finestra corrente è puntata da CWP (Current Window Pointer). Dell'atto di una chiamata: CWP viene decrementato; La finestra trasla di 16 registri; I parametri di output diventano parametri in input; lo stack pointer diventa il frame pointer → tutto ciò con una semplice ridenominazione e grazie ad una grossa disponibilità di registri

LIVELLO ISA DELL'8051: utilizzato per applicazioni embedded. Ha una bassa modularità e nessun tipo di protezione

- 8 registri ufficiali da 8 bit
- Unico spazio di memoria con: REGISTRI UFFICIALI; MEMORIA INDIRIZZABILE AL BIT; REGISTRI SPECIALIZZATI (PSW, gestione interrupt e timer)
  Il metodo di ridenominazione è frequentemente utilizzato
  Il registro basso fino a 127 sono i registri del microp. mentre gli altri sono della memoria principale:

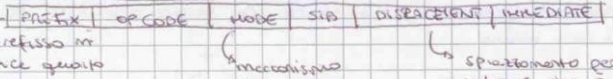


- FORMATO DELLE ISTRUZIONI -



Codice operativo: contiene il tipo di operazione che si vuole fare, con un massimo di 3 operandi. Il codice op decodifica l'operazione
I campi indirizzo possono fare riferimento sia alla memoria che ai registri
→ LUNGHEZZA DELLE ISTRUZIONI: Ci sono scelte diverse a seconda di in quanto possono essere o fisse o variabili. Quello FISSA semplifica le FETCH delle istr.
Una word può contenere uno o più istruzioni. Se le istr. hanno formato variabile → dove non c'è relazione tra word e istruzioni
  - Il numero di bit dedicati al COD. OPERATIVO non è costante
  - Alcune porzioni comuni del codice operativo non ci devono essere interruzioni

FORMATO ISTRUZIONI PENTIUM IV:



- PREFISSO: indica quando è lungo l'istruzione
- MODE: meccanismo di indirizzamento
- SIB: spiazzamento per indicare che è richiesto per segmento (OFFSET) (indirizzo all'interno del segmento)
- IMMEDIATE: è valore usato dove che è già presente nelle istruzioni macchine

FORMATO DI ISTRUZIONI SPARC: ha lunghezza fissa a 32bit delle versione base
Istruzioni aritmetiche a 3 indirizzi SRC1, SRC2, DEST
- Disponibili solo 4 formati distinguibili dai primi 2 bit → mi dice in quale gruppo sto
- I 32 registri della finestra sono indirizzati con 5 bit

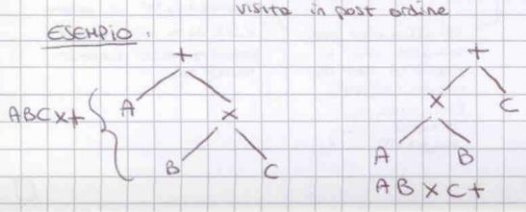
FORMATO DELLE ISTRUZIONI 8051: istruzioni variabili da 1, 2 o 3byte
Il massimo n° di operandi è pari a 2. Quest'architettura ha il registro accumulatore e quindi è inutile introdurre il 3° operando

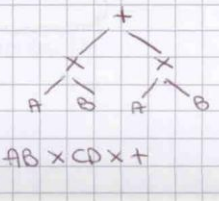
MODALITÀ DI INDIRIZZAMENTO

A MEMORIA PRINCIPALE
- IMMEDIATO: il valore dell'operando è nell'istruzione macchina tipicamente una costante
- DIRETTO: l'istruzione contiene l'indirizzo di memoria completo dell'operando (in memoria principale)
- INDIRETTO: l'indirizzo di memoria fornito contiene l'indirizzo dell'operando (concetto di puntatore)
- A REGISTRO: si specifica un registro che contiene l'operando (o che ha ricevuto)
- INDIRETTO A REGISTRO: il registro specificato contiene l'indirizzo dell'operando
- INDICIZZATO: l'indirizzo è dato da una costante più il contenuto di un registro (è il punto di partenza) contiene l'indice
- A REGISTRO BASE: l'indirizzo del primo elemento dell'array di uno un registro che contiene l'indice viene sommato a tutti gli indirizzi, il contenuto di un indirizzo registro
- A STACK: l'operando è sulla cima dello stack → si può accedere solamente al registro offset non ho bisogno di indicare la specifica dell'indirizzo

ES: INDIRIZZAMENTO INDIZZATO A REGISTRO
```
MOV R1, #0;   // indirizzamento immediato poiché # costante
MOV R2, #A;
MOV R3, #A+1024;  // sommo A con 1024 in R3 via a finire poiché ogni elemento è di 1byte
LOOP:   meccanismo di indirizzamento indiretto avviene in R2 c'è l'indirizzo
ADD R2, #4
CMP R2, R3;
BLT LOOP;  Branch if then less se R2, è minore di R3 // implementato la somma dei segmenti dell'array
```
ES: INDIRIZZAMENTO INDICIZZATO:
```
MOV R1, #0   // indirizz. a reg immediato
MOV R2, #0
MOV R3, #4096
LOOP: MOV R4, A(R2)   // metto in R4 il contenuto l'esimo di R2
AND R4, B(R2)
OR R1, R4
ADD R2, #4    // incremento di 4 poiché è un byte per passare dell'indirizzo successivo
CMP R2, R3
BLT LOOP    "calcola OR di A[i] AND B[i] dove A e B sono due array
```
→ INDIRIZZAMENTO INDICIZZATO ESTESO: Indirizzo di memoria è calcolato sommando il contenuto di due registri: ① un registro memorizza la base ② un registro memorizza l'indice
```
LOOP: MOV R4, (R2+R5)   es. interpreto come indirizzo di memoria
AND R4, (R2+R6)
```
INDIRIZZAMENTO A STACK: è utilizzato per gestire le procedure, calcolare espressioni aritmetiche, oppure gestire interrupt. Si può accedere solamente all'indirizzo offrendo
→ OPERAZIONI: PUSH: aggiunge un elemento alla cima dello stack; POP: preleva un elemento dalla cima dello stack } operazioni aritmetiche su due elementi offerti che mettono al loro posto il risultato
→ NOTAZIONE POLACCA INVERSA: per calcolare un'espressione algebrica usando lo stack, occorre convertirla dalla forma infissa alla forma postfissa
- FORMA INFISSA: l'operatore è scritto tra gli operandi e necessita di parentesi
- FORMA POST-FISSA: l'operatore segue gli operandi e non richiede parentesi
- Rappresentando l'espressione con un albero, la forma post-fissa si ottiene tramite una visita in post ordine

ESEMPIO:





Sorgente : http://holsi.hasanaj.com/
Autore : Holsi Hasanaj
Professore: Riccardo Torlone

Corso di Ingegneria Informatica Roma Tre
Anno di produzione: 2011-2012

LIVELLO DELLE ISTRUZIONI MACCHINA

CALCOLO ESPRESSIONI: BIPUSH aggiunge l'operando sulla cima dello STACK e incrementa SP di 1. IADD somma gli elementi affioranti. IMUL moltiplica gli elementi affioranti → li rimpiazzano con il risultato e decrementano di 1 SP

- Questa procedura è molto sicura in quanto non si usano i registri ma tutte le operazioni vengono fatte sul registro di STACK

ORTOGONALITÀ: un set di istruzioni è caratterizzato da: ① Codici Operativi ② Modalità di indirizzamento → in genere non tutte modalità di indirizzamento sono utilizzabili con tutti i codici operativi.
→ Se questo avviene si dice che i codici e le modalità di indirizzamento sono tra loro ortogonali. L'ortogonalità è una proprietà molto desiderabile perché semplifica la generazione del codice

INDIRIZZAMENTO PENTIUM IV: Uno degli operandi è sempre un registro specificato dal campo REG dell'istruzione → l'altro è specificato da MOD e R/M. → 32 indir. diversi

- MOD = 00 indirizzamento tramite registro, tramite R/M si sceglie quale registro
- MOD = 01 ripete le stesse modalità con OFFSET a 8 bit (in coda all'istruzione)
- MOD = 10 ripete le stesse modalità con OFFSET a 32 bit " "
- MOD = 11 viene utilizzato se il secondo operando è un registro: a 32 bit per le istruz. a word, a 8 per quelle a byte

→ SIB (Scale Index Base) tipologia di indirizzamento che specifica ① Fattore di SCALA ② Registro indice (EAX) ③ Registro base (EBP) → indirizz. viene calcolato come BASE + INDEX·SCALE + OFF

INDIRIZZAMENTO ULTRASPARC: indirizzamento immediato o a registri per tutte le istruzioni aritmetiche e logiche. → 5 bit indirizzano i 32 registri della finestra
Solo LOAD e STORE indirizzano la memoria → ci sono anche 2 ulteriori modi di indirizzamento a memoria ① tramite somma di due registri ② tramite registro indice con OFFSET a 13 bit

INDIRIZZAMENTO 8051: sono semplici e regolari: Indirizzamento implicito dell'accumulatore, Indirizzamento a registro, Indirizzamento diretto, Indiretto a registro, Immediato

TIPI DI ISTRUZIONI MACCHINA:

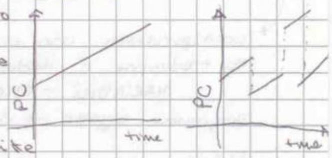
- Istruzioni di movimento: MOV spostano dei dati da un registro ad un altro attraverso la copia
- Istruzioni binarie: hanno 2 operandi e producono un risultato che può andare sullo stack locato su un'altr
- Istruzioni UNARIE: prendono un risultato e lo shiftano
- SALTI: possono essere condizionali e non, servono a codificare istruzioni condiz. e cicli
- Chiamate di procedure: alterano il flusso di esecuzione attraverso una gestione "A PILA"

→ ISTRUZIONI I/O: operaz. I/O consiste nel trasferimento di dati tra un device d'IO e la memoria

1) I/O programmata con busy waiting: la CPU interroga periodicamente i dispositivi (polling) e cicla a vuoto
2) I/O gestito con interruzioni: la CPU effettua i trasferimenti con la memoria e avvia l'operazione di I/O ma si dedica ad altro fino a che il device non manda un interrupt
3) DMA (direct Memory Access) la CPU avvia l'operazione poi gestita integralmente dal DMA. Inoltre il controller DMA può gestire più operazioni contemporaneamente

CONTROLLO DEL FLUSSO: tecniche che possono alterare l'esecuzione

- CHIAMATA DI PROCEDURA: per ciascuna chiamata viene allocato sullo stack, uno nuovo stack frame
- Lo STACK FRAME contiene: ① I parametri in entrata e in uscita ② le variabili locali ③ l'indirizzo di rientro ④ un puntatore allo STACK frame del chiamante
  - l'accesso ai parametri e alle variabili locali avviene tramite offset da BP



- COROUTINE: due ROUTINE (processi) vengono eseguiti contemporaneamente con un meccanismo di RESUME, combinando esecuzioni di istruzioni di un processo con l'altro. Alta simmetria → dovuto all'esecuzione alternata. Ciascuna RESUME riparte dall'istruzione successiva alla precedente

TRAP : è una procedura automatica che viene innescata da una condizione eccezionale che si verifica durante l'esecuzione di un programma. Le trap sono sincrone e dipendono da quello che succede sulla CPU → GESTORE DELLA TRAP nel S.O che cerca di risolvere
ES: → OVERFLOW / VIOLAZIONE DELLA PROTEZIONE / DIVISIONE PER 0
INTERRUZIONI: interruzione del flusso delle istruzioni → qualcosa che avviene durante l'esecuzione del programma → esterno alla procedura

ARCHITETTURA IA-64: l'intel è orientata a competere con l'IA-32. L'IA-64 è un architetture a 64 bit sviluppata in collaborazione con HP. L'idea alla base → spostare il carico di lavoro dall'esecutore alla compilazione
- GLI operandi possono essere a 64bit (sempre lo stesso ingombro)
- MICROPROC ITANIUM: completamente RISC perché forza altri proc. → sposta il lavoro a monte di esecuzione
- si suppone di avere tantissimi registri 128 generali purpose 128 in virgola mobile

→ SCHEDULING ISTR. 64: si cerca di eliminare le possibilità di esecuzione parallela delle istruz. supportate. Le istr. vengono in BUNDLE: pacchetti di 3 senza dipendenze. La parte di TEMPLATE fornisce informazioni sulla possibilità di eseguire parallela su unità funzionali indipendenti

PROBL. IA32: è immediatamente CISC, le sue istruzioni possono essere spezzate in istruzioni RISC, ma questo richiede tempo e spazio su chip
- pochi registri ~~segmento~~ ed asimmetrici: con molte dipendenze rende difficile l'esecuzione parallela di più istruzioni
- 4GB di spazio di indirizzamento

SCHEDULING: pianificazione durante l'esecuzione dei tempi di

→ ESECUZIONE PREDICATIVA: si cerca di diminuire i salti condizionali con la tecnica della esecuzione predicativa → estensione del concetto di esecuzione condizionale
CMOVZ R2, R3, R1 → fa lo spostamento solamente se R1 è 0 → caso visto con l'hardware durante il corso
- nella versione condizionale, il codice è costituito da un unico blocco basico senza salti
- L'unico vincolo per l'esecuzione di ciascuna istruzione è la conoscenza della condizione
(1) istruzioni predicative possono andare in pipeline senza problemi

CAP 7

LINGUAGGI ASSEMBLATIVI

I linguaggi ad alto livello generalmente offrono migliori prestazioni a livello di codice, tuttavia nella programmazione bisogna utilizzare un approccio misto: ovvero isolare le parti critiche e codificare solo queste in linguaggio assembleativo
VANTAGGI ASSEMBLY LANGUAGE: può servire per ottimizzare una applicazione (approx mixed) → a volte è l'unica risorsa disponibile, ad esempio EMBEDDED. È molto utile alla comprensione dell'architettura di un calcolatore

TRADUZIONE: PROGRAMMA SORGENTE → TRADUTTORE (• Compilatore • Assemblatore) → PROGRAMMA OGGETTO

Il programma sorgente è scritto in un linguaggio simbolico, orientato all'utente
Il programma oggetto ① esegue le operazioni specificate dal programma sorgente ② è scritto in linguaggio macchina e quindi direttamente eseguibile
→ costituisce i dati del programma traduttore, il programma oggetto ne è il risultato

INTERPRETAZIONE esegue le istruzioni di programma man mano che si esegue il programma.
→ viene effettuata ogni volta che il programma viene eseguito poiché non viene generato nessun programma oggetto. L'interprete esegue direttamente le istruzioni del programma sorgente. ES: SQL, PROLOG
P. SORGENTE → INTERPRETE → SIST DI ELABOR EVE del S.O

corrispondenza uno a uno con le istruzioni macchina. Il traduttore è detto ASSEMBLATORE e la traduzione è detta assemblaggio. Esistono delle PSEUDOISTRUZIONI, per fare delle dichiarazioni
VARIABILI = ETICHETTA (più a basso livello) → ad esse non corrispondono istruzioni nel programma oggetto → dichiarano etichette costrutti ed esprimono direttive di assemblaggio

MACRO DEFINIZIONI: associano un nome ad un segmento di codice; il nome così definito può essere usato più volte nel corso del programma (compilatore) → La sostituzione della macro avviene staticamente in fase di assemblaggio

Calcolatori Elettronici — ARCHITETTURA CALCOLATORI

LINGUAGGI ASSEMBLATIVI — CAP 7 — 17/

MACRO DEFINIZIONI: ex → associano un nome simbolico a più istruzioni

```
ex:  MOV  EAX, P          SWAP  MACRO
     MOV  EBX, Q                MOV EAX,P
     MOV  Q, EAX                MOV EBX,Q
     MOV  P, EBX                MOV Q, EAX
                                MOV P, EBX
                                ENDM
                                SWAP
```

MACRO CON PARAMETRI: la sostituzione dei parametri formali con gli argomenti attuali viene effettuata staticamente in fase di assemblaggio. A valle dell'assemblaggio, a sostituzione avvenuta, nel programma oggetto non resta nessuna traccia della macro

```
ex:  MOV  EAX, P          CHANGE  MACRO P1, P2
     MOV  EBX, Q                  MOV EAX, P1
     MOV  Q, EAX                  MOV EBX, P2
     MOV  P, EBX                  MOV P2, EAX
                                  MOV P1, EBX
     MOV  EAX, R                  END M
     MOV  EBX, S                  CHANGE P,Q
     MOV  S, EAX                  CHANGE R,S
     MOV  R, EBX
```

DIFFERENZE MACRO E PROCEDURE: la chiamata della macro avviene durante l'assemblaggio mentre quella di procedura durante l'esecuzione del programma.
Il corpo della macro è inserito dentro nel programma oggetto mentre nella procedura la procedura di invocazione spunta nel caso della macro mentre nella procedura no
Il tipo di ritorno non è presente per la macro mentre per la procedura sì
Quante copie compaiono nel corpo del programma oggetto? Una per chiamata di macro mentre una sola per la procedura

PROCESSO DI ASSEMBLAGGIO: l'assemblatore traduce da linguaggio assemblativo a linguaggio macchina. Processo molto semplice salvo per le istruzioni con riferimenti a (SALTI)

→ ASSEMBLATORE AD UNA PASSATA: ① Traducono in un formato intermedio compatto ② Mettono la forma intermedia in una tabella ③ Alla fine traduce il contenuto della tabella

→ ASSEMBLATORE A 2 PASSATE: - PRIMA PASSATA: ① Calcola la lunghezza e la posizione delle istruzioni ② Genera una tavola dei SIMBOLI
- SECONDA PASSATA: ① Tutti i riferimenti sono noti nella tavola dei simboli ② Genera il codice usando i valori della tavola

PRIMA PASSATA: valuta la eventuale etichetta di ciascuna istruzione, calcola il valore dell'ILC (Instruction Location Counter). Inserisce tutte le etichette nella tabella dei simboli con i corrispondenti valori di ILC → alla fine della passata i riferimenti in avanti sono noti

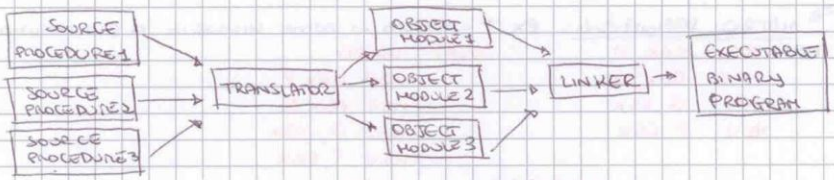
SECONDA PASSATA: sfrutta il prodotto della prima passata, trasforma i riferimenti corretti, la tavola dei simboli fornisce l'indirizzo di tutte le etichette.
Genera il codice, istruzione per istruzione, traducendo la tavola dei codici operativi che fornisce: ① La lunghezza delle istruzioni ② I codici operativi ③ La codifica delle istruzioni che dipende dal tipo di operandi e dalle modalità di indirizzamento

→ ORGANIZZAZIONE DELLE TABELLE DEI SIMBOLI: → tabella che mi dica per ciascuna etichetta qual è l'indirizzo dell'istruzione che ha quella etichetta.
ESEMPIO → funzione di HASH che prende una stringa che attraverso un algoritmo genera una tabella di HASH → questa tabella ha un numero di elementi maggiore delle etichette quindi due non possono collidere

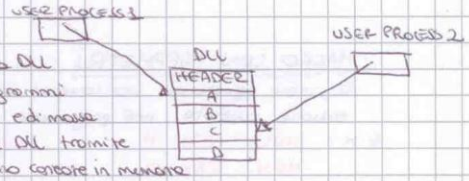
→ COLLEGAMENTO E CARICAMENTO: mette insieme i vari moduli per produrre il programma eseguibile → produce un unico modulo assoluto pronto ad essere caricato

COLLEGAMENTO: Ciascun modulo ha il suo spazio di indirizzi. Quando i moduli vengono collegati occorre ① traslare i loro spazi degli indirizzi ② risolvere tutti i riferimenti esterni

CARICAMENTO: (LOADER) definisce gli indirizzi finali e carica il modulo risultato in memoria centrale



DYNAMIC LINK LIBRARY (DLL)



È una soluzione adottata in ambiente WINDOWS. Una DLL può contenere sia procedure che dati condivisi più programmi condividono una DLL. Risparmio di memoria centrale ed di massa

- IMPLICIT LINKING: il programma è collegato alla DLL tramite una tabella, la DLL necessaria viene caricata in memoria
- EXPLICIT LINKING: la DLL viene richiesta e caricata dinamicamente all'atto della chiamata

LINGUAGGIO ASSEMBLATIVO i 386

Il linguaggio assembleativo (assembly): rappresentazione simbolica dell'insieme di istruzioni macchina di un'architettura. Associa ai dati nomi simbolici che identificano le corrispondenti posizioni in memoria. → Nasconde i dettagli relativi alle singole istruzioni (codici operativi, formati non quelli relativi all'architettura della macchina

→ Si scrive un linguaggio ASSEMBLY per conoscere meglio il calcolatore: ① Per imparare l'assembler bisogna conoscere l'architettura ② Il debug dell'assembler ci fa comprendere come funziona l'architettura

→ Visto che generalmente bisogna scrivere frammenti di codice per ricoprire solamente la parte critica del programma

Ex: ASSEMBLER "EMBEDDED"

```
#include <stdio.h>
void main (void) {
static int X = 3,
asm { MOV %EAX, X
ADD %EAX, X
ADD %EAX, X
NOP
MOV X, %EAX }
PRINTF ("%d\n", X);
return; }
```

ORGANI FUNZIONALI ASSEMBLY

1) ASSEMBLATORE: programma che riceve in ingresso un programma in linguaggio assembleativo e genera un programma in linguaggio macchina (binario) pronto per essere eseguito dall'HARDWARE
2) TRACER: è un simulatore dell'esecuzione di un programma scritto in un linguaggio assembleativo. Consente di procedere "PASSO-PASSO". Debugger per assembler.

ASSEMBLER 8088: i 386 per x86 → linguaggio → processo

È una versione semplificata di un micro processore Intel moderno → tuttavia le codici restano assembler può essere eseguito sui moderni calcolatori

→ CARATTERISTICHE MICROPROCESSORE: Architettura: a 16 bit; Address Bus: 20 bit; Data bus: 8 bit; Unità indirizzabile: 1 byte

REGISTRI:

REGISTRI DI USO GENERALE:

(AX): registro accumulatore, usato per memorizzare il risultato dell'elaborazione e come dati nazione di molte istruzioni
(BX): registro base, usato come accumulatore o come puntatore alla memoria
(CX): registro contatore usato come contatore di cicli
(DX): registro dati usato insieme ad AX per estendere fino a 32 bit

	GENERAL REGISTER			GENERAL REGISTER	
AX	AH	AL	CX	CH	CL
BX	BH	BL	DX	DH	DL
	8 bit	8 bit		8 bit	8 bit

Calcolatori Elettronici — ARCHITETTURA CALCOLATORI

CAP 4 18/

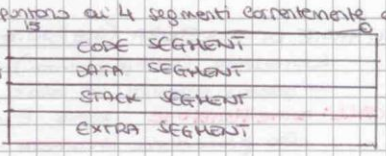
REGISTRI

REGISTRI PUNTATORI: (SP) registro puntatore alla cima dello stack → viene modificato automaticamente dalle operazioni sullo stack PUSH, POP
(BP): registro puntatore base dello stack → punta alla base del frame (record di attivazione) assegnato alla procedura corrente
(SI): registro indice sorgente, usato in combinazione con BP per riferirsi a dati sullo stack o con BX per accelerare i dati in memoria
(DI): registro indice destinazione usato come SI

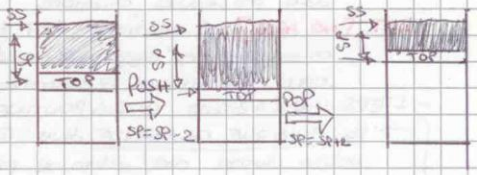
REGISTRO DI STATO: il registro di stato (Flag) è un insieme di registri da 1 bit. I bit sono impostati da istruzioni aritmetiche
(Z) il risultato è zero (S): il risultato è negativo (V) il risultato ha causato un OVERFLOW
(C) il risultato ha riportato ha generato un riporto (A) riporto ausiliario (P) parità del risultato
→ Gli altri bit del registro controllano alcuni aspetti dell'attività del processore:
(I) attività di Interrupt (T) abilita il tracing (D) Operazioni sulle stringhe

SEGMENTAZIONE DELLA MEMORIA: Lo spazio di memoria viene visto come un gruppo di segmenti. Ogni segmento costituisce un'unità di memoria indipendente è formata da locazioni contigue di memoria → ha un limite MAX di 64 KB → ha un indirizzo di memoria multiplo di 16. Il sistema operativo mi fissa la base quindi c'è l'indirizzo effettivo si ha con il valore + somma segmento

REGISTRI DI SEGMENTO: i registri di segmento puntano ai 4 segmenti correntemente attivi
CS: punta al segmento contenente le istr. da eseguire
DS: punta al segmento contenente le variabili del pg
SS: Punta al segmento contenente lo stack corrente
ES: punta al segmento EXTRA, usato tipicamente per i dati



SEGMENTO PER LA GESTIONE DELLO STACK: lo stack cresce andando dagli indirizzi alti a quelli bassi. SS punta all'indirizzo di partenza dello stack (SP) punta alla locazione in cima allo stack

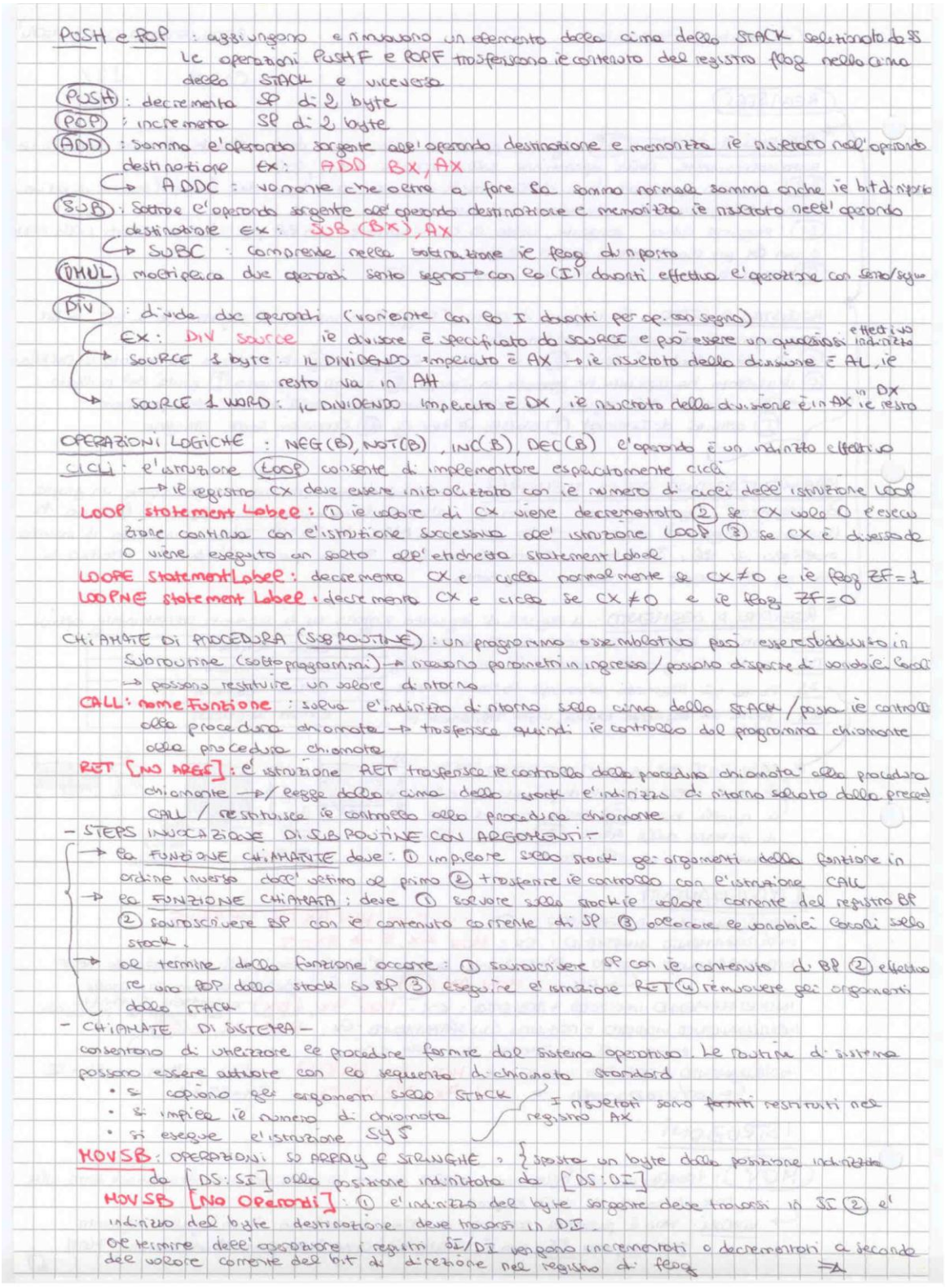


INDIRIZZAMENTI:

INDIRIZZAMENTO A REGISTRO: EX: CX=5 → MOV AX, CX → AX=5
INDIRIZZAMENTO IMMEDIATO: EX: MOV AX, 5 → AX=5
INDIRIZZAMENTO DIRETTO: l'istruzione contiene l'indirizzo dei dati nell'operando stesso
↳ EX: MOV AX, (4) → è l'indirizzo di mem. principale
INDIRIZZAMENTO INDIRETTO A REGISTRO: EX: MOV AX, (BX) → contiene l'indirizzo di memoria principale
INDIRIZZAMENTO INDIRETTO A REGISTRO CON SPIAZZAMENTO: Ex: MOV AX, 2(SI)
prendo SI ci sommo 2 ⇒ 10
INDIRIZZAMENTO A REGISTRO INDICE: EX: MOV AX, (BX)(SI) e' indirizzo si ottiene con BX + SI
↳ con fattore di scalamento → MOV AX, 2(BX)(SI) BX+SI+2

ISTRUZIONI:

MOV: trasferisce un byte o una WORD da una sorgente ad una destinazione tenta alle regole che il contenuto della sorgente
↳ VINCOLI: non è possibile caricare un valore immediato in un registro segmento
Il registro CS non è utilizzabile come destinazione di un'istruzione MOV



PUSH e POP : aggiungono e rimuovono un elemento dalla cima dello STACK selezionato da SS
Le operazioni PUSHF e POPF trasferiscono il contenuto del registro flag nella cima dello STACK e viceversa
PUSH : decrementa SP di 2 byte
POP : incrementa SP di 2 byte
ADD : somma l'operando sorgente all'operando destinazione e memorizza il risultato nell'operando destinazione ex: ADD BX, AX
→ ADDC : uguale che oltre a fare la somma normale somma anche il bit di riporto
SUB : sottrae l'operando sorgente all'operando destinazione e memorizza il risultato nell'operando destinazione ex: SUB (BX), AX
→ SUBC : comprende nella sottrazione il flag di riporto
IMUL moltiplica due operandi senza segno → con ea (I) davanti effettua l'operazione con segno/sgn
DIV : divide due operandi (versione con ea I davanti per op. con segno)
ex: DIV SOURCE il divisore è specificato da SOURCE e può essere un qualsiasi effettivo indirizzo
SOURCE 1 byte : il DIVIDENDO implicito è AX → il risultato della divisione è AL, il resto va in AH
SOURCE 1 WORD : il DIVIDENDO implicito è DX, il risultato della divisione è in AX in DX il resto
OPERAZIONI LOGICHE : NEG(B), NOT(B), INC(B), DEC(B) l'operando è un indirizzo effettivo
CICLI : l'istruzione LOOP consente di implementare esplicitamente cicli
→ il registro CX deve essere inizializzato con il numero di cicli dell'istruzione LOOP
LOOP statement Label : ① il valore di CX viene decrementato ② se CX vale 0 l'esecuzione continua con l'istruzione successiva dell'istruzione LOOP ③ se CX è diverso da 0 viene eseguito un salto all'etichetta statement Label
LOOPE statement Label : decrementa CX e cicla normalmente se CX≠0 e il flag ZF=1
LOOPNE statement Label : decrementa CX e cicla se CX≠0 e il flag ZF=0
CHIAMATE DI PROCEDURA (SUBROUTINE) : un programma assemblativo può essere suddiviso in Subroutine (sottoprogrammi) → possono parametri in ingresso / possono disporre di variabili locali
→ possono restituire un valore di ritorno
CALL: nomeFunzione : salva l'indirizzo di ritorno sulla cima dello STACK / passa il controllo alla procedura chiamata → trasferisce quindi il controllo dal programma chiamante alla procedura chiamata
RET [NO ARGS] : l'istruzione RET trasferisce il controllo della procedura chiamata alla procedura chiamante → legge dalla cima dello stack l'indirizzo di ritorno salvato dalla precedente CALL / restituisce il controllo alla procedura chiamante
- STEPS INVOCAZIONE DI SUBROUTINE CON ARGOMENTI -
→ La FUNZIONE CHIAMANTE deve : ① impilare sullo stack gli argomenti della funzione in ordine inverso dall'ultimo al primo ② trasferire il controllo con l'istruzione CALL
→ La FUNZIONE CHIAMATA : deve ① salvare sullo stack il valore corrente del registro BP ② sovrascrivere BP con il contenuto corrente di SP ③ allocare le variabili locali sullo stack
→ al termine della funzione occorre : ① sovrascrivere SP con il contenuto di BP ② effettuare una POP dallo stack su BP ③ eseguire l'istruzione RET ④ rimuovere gli argomenti dallo STACK
- CHIAMATE DI SISTEMA -
consentono di utilizzare le procedure fornite dal sistema operativo. Le routine di sistema possono essere attivate con la sequenza di chiamata standard
- si copiano gli argomenti sullo STACK
- si impila il numero di chiamata
- si esegue l'istruzione SYS
I risultati sono forniti restituiti nel registro AX
MOVSB : OPERAZIONI SU ARRAY E STRINGHE : sposta un byte dalla posizione indirizzata da [DS:SI] alla posizione indirizzata da [DS:DI]
MOVSB [NO Operandi] : ① l'indirizzo del byte sorgente deve trovarsi in SI ② l'indirizzo del byte destinazione deve trovarsi in DI.
Al termine dell'operazione i registri SI/DI vengono incrementati o decrementati a seconda del valore corrente del bit di direzione nel registro di flag

Calcolatori Elettronici — ARCHITETTURA CALCOLATORI CAP 6 dispensa 9 — 19/

MOVSB [No Operandi]:
Al termine dell'operazione i registri SI/DI vengono incrementati o decrementati a secondo del valore corrente del bit di direzione nel registro di FLAG
CLD: con questa istruzione i registri SI e DI vengono incrementati (scorrimento avanti)
STD: con questa istruzione, i registri SI e DI vengono decrementati (scorrim. indietro)

REP MOVSB: itera l'esecuzione dell'istruzione MOVSB un numero di volte pari al valore corrente del registro CX

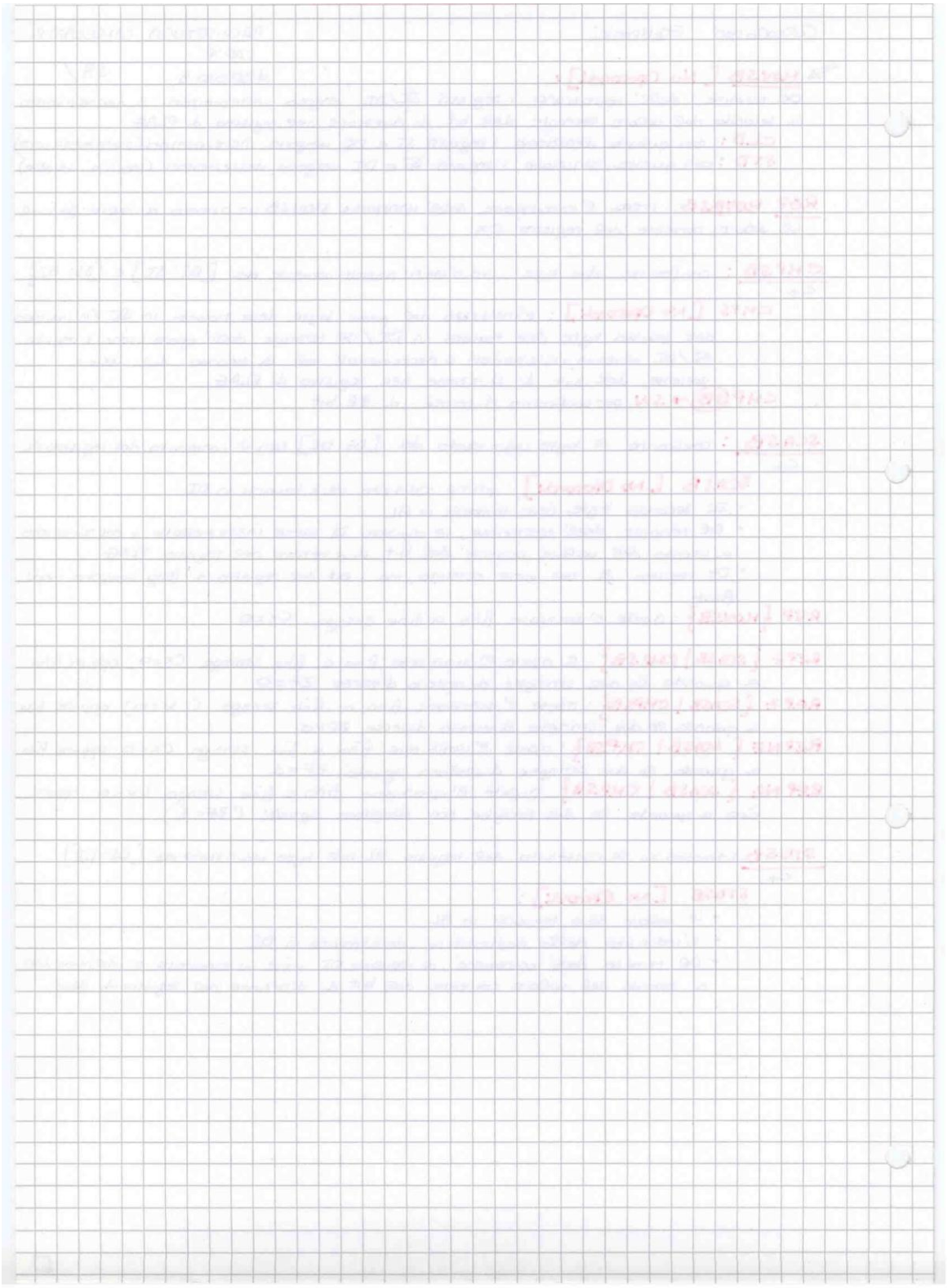
CMPSB: confronta due byte, indirizzati rispettivamente da [DS:SI] e [DS:DI]
↳
CMPS [No Operandi]: l'indirizzo del primo byte deve trovarsi in SI / l'indirizzo del secondo byte deve trovarsi in DI / Al termine dell'operazione i registri SI/DI vengono incrementati o decrementati a secondo del valore corrente del bit di direzione nel registro di FLAG
CMP(SB) → SW per confronto di parole di 16 bit

SCASB: confronta il byte indirizzato da [DS:DI] con il contenuto del registro AL
↳
SCASB [No Operandi]:
- Il secondo byte deve trovarsi in AL
- Al termine dell'operazione, il registro DI viene incrementato o decrementato a secondo del valore corrente del bit di direzione nel registro FLAG
- Il registro AL non viene alterato, ma i bit del registro di flag vengono modificati

REP {MOVSB}: ripete l'istruzione fino a fine stringa CX=0

REPE {SCASB | CMPSB}: ripete l'istruzione fino a fine stringa CX=0 oppure fino a quando le due stringhe diventano diverse ZF=0
REPZ {SCASB | CMPSB}: ripete l'istruzione fino a fine stringa (CX=0) oppure fino a quando le due stringhe diventano diverse ZF=0
REPNE {SCASB | CMPSB}: ripete l'istruzione fino a fine stringa CX=0 oppure fino a quando le due stringhe diventano uguali ZF=1
REPNZ {SCASB | CMPSB}: ripete l'istruzione fino a fine stringa CX=0 oppure fino a quando le due stringhe non diventano uguali (ZF=1)

STOSB: trasferisce il contenuto del registro AL nel byte indirizzato da [DS:DI]
↳
STOSB [No Operandi]:
- il valore deve trovarsi in AL
- l'indirizzo della destinazione deve trovarsi in DI
- Al termine dell'operazione, il registro DI viene incrementato o decrementato a secondo del valore corrente del bit di direzione nel registro di flag



Calcolatori Elettronici — Architettura Calcolatore dispensa 10

ASSEMBLATORE ASSB — CAP 7 19/

Direttive del compilatore → OGNI PROGRAMMA ASSEMBLY deve essere strutturato in 3 sezioni:
1) Sezione di TESTO, contiene le istruzioni del programma (direttiva: .SECT.TEXT)
2) sezione DATI alloca spazio nel segmento DATI per i dati inizializzati (direttiva .SECT.DATA)
3) Sezione BSS alloca spazio nel segmento dati (non inizializzato) direttiva .SECT.BSS

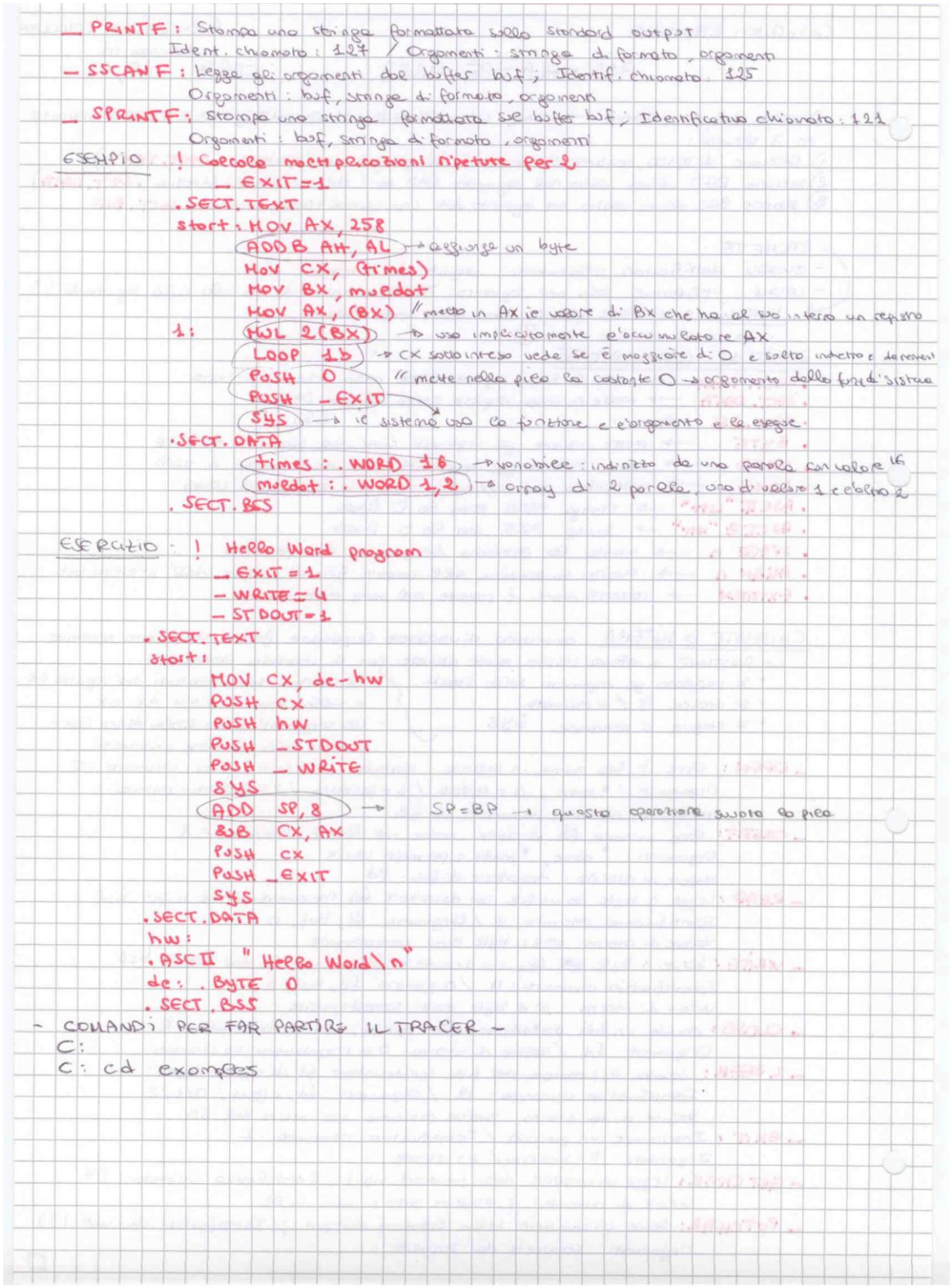
ETICHETTE:
- GLOBALI: identificatori alfanumerici seguiti dal simbolo ":"
- LOCALI: utilizzabili solo nel segmento TESTO, costituite da uno solo cifra seguito da ":"
  → VINCOLI: le etichette globali devono essere univoche
  I commenti iniziano con (!)

DIRETTIVE COMPILATORE
- .SECT.TEXT → mette le linee seguenti nella sezione di testo
- .SECT.DATA → mette le linee seguenti nella sezione DATA
- .SECT.BSS → " " " " BSS
- .BYTE → mette insieme gli argomenti come una sequenza di BYTE
- .WORD → mette insieme gli argomenti come una sequenza di WORD
- .LONG → assembla gli argomenti come una sequenza di LONG
- .ASCII "stringa" → stringa ASCII senza lo 0 finale
- .ASCIZ "stringa" → stringa ASCII con lo 0 finale
- .SPACE n → avanza la posizione del puntatore di n posizioni
- .ALIGN n → avanza la posizione del contatore fino al confine dell'n-esimo byte
- .EXTERN → identificatore è presente nel nome ma esterno

CHIAMATE DI SISTEMA: consentono di utilizzare le procedure fornite dal sistema operativo
Le ROUTINE di sistema possono essere attivate con la chiamata standard:
- Si impilano gli argomenti sullo stack
- Si impila il n° di chiamata
- Si esegue l'istruzione SYS

→ - I risultati sono restituiti nel registro AX o nella combinazione AX:DX
- Gli argomenti sullo STACK devono essere rimossi dalla funzione chiamante

- OPEN: Apre il file nome in lettura - scrittura → Identificativo chiamata: 5
  Argomenti: *nome, 0 = lettura / 1 = scrittura / 2 = lettura - scrittura
  Valore di ritorno: descrittore di file
- CREAT: Crea un nuovo file di nome nome → Identif. chiamata = 8
  Argomenti: *nome, *mode = permessi UNIX
  Valore di ritorno: descrittore di file fd
- READ: Legge n byte da un file con descrittore fd trasferendoli nel buffer buf
  Identificativo chiamata: 3 / Argomenti: fd, buf, n
  Valore di ritorno: n° di byte letti correttamente
- WRITE: Scrive n byte sul file con descrittore fd prelevandoli dal buffer buf
  Identificativo chiamata: 4 / Argomenti fd, buf, n;
  Valore di ritorno: n° di byte scritti correttamente
- CLOSE: chiude un file precedentemente aperto; Identif. chiamata: 6
  Argomenti: fd / Valore di ritorno: 0 se l'operazione ha successo
- LSEEK: Sposta il puntatore del file con descrittore fd di offset bytes
  Identificativo chiamata: 19 / Argomenti: fd, offset, 0/1/2
  Valore di ritorno: nuova posizione all'interno del file
- EXIT: Interrompe un processo / Identificativo chiamata: 1
  Argomenti: 0 = Successo / 1 = errore
- GETCHAR: Legge un carattere dallo standard input; Identificativo chiamata: 117
  Valore di ritorno: il carattere letto è posto in AL
- PUTCHAR: Scrive un carattere sullo standard output; Identificativo chiamata 122
  Argomenti: carattere da scrivere



- PRINTF: Stampa una stringa formattata sullo standard output
Ident. chiamato: 127 / Argomenti: stringa di formato, argomenti
- SSCANF: Legge gli argomenti dal buffer buf; Identif. chiamato: 125
Argomenti: buf, stringa di formato, argomenti
- SPRINTF: Stampa una stringa formattata sul buffer buf; Identificativo chiamato: 121
Argomenti: buf, stringa di formato, argomenti

ESEMPIO

```
! Calcola mediante iterazioni ripetute per 2
    _EXIT=1
.SECT.TEXT
start: MOV AX,258
       ADDB AH,AL      → aggiunge un byte
       MOV CX,(times)
       MOV BX,mydat
       MOV AX,(BX)     // mette in AX il valore di BX che ha al suo interno un registro
1:     IMUL 2(BX)      → uso implicitamente l'accumulatore AX
       LOOP 1b         → CX sottointeso vede se è maggiore di 0 e salto indietro e decrementa
       PUSH 0          // mette nella pila la costante 0 → argomento della fine del sistema
       PUSH _EXIT
       SYS             → il sistema usa la funzione e l'argomento e la esegue.
.SECT.DATA
       times: .WORD 16     → variabile: indirizzo di una parola con valore 16
       mydat: .WORD 1,2    → array di 2 parole, una di valore 1 e l'altra 2
.SECT.BSS
```

ESERCIZIO

```
! Hello Word program
    _EXIT = 1
    _WRITE = 4
    _STDOUT = 1
.SECT.TEXT
start:
    MOV CX,de-hw
    PUSH CX
    PUSH hw
    PUSH _STDOUT
    PUSH _WRITE
    SYS
    ADD SP,8        → SP=BP → questa operazione svuota la pila
    SUB CX,AX
    PUSH CX
    PUSH _EXIT
    SYS
.SECT.DATA
hw:
.ASCII "Hello World\n"
de: .BYTE 0
.SECT.BSS
```

- COMANDI PER FAR PARTIRE IL TRACER -

```
C:
C: cd examples
```