



ISTITUTO STATALE DI ISTRUZIONE SUPERIORE

“VITTORIO VENETO” Città della Vittoria

Via Vittorio Emanuele II, 70 - 31029 VITTORIO VENETO (TV) tel. 0438-57147

PROGRAMMA SVOLTO

INSEGNANTE: Florindo Bassi

Materia d'insegnamento: Tecnologie e Progettazione di Sistemi Informatici e di Telecomunicazioni

CLASSE 3°B inf ITT

ANNO SCOLASTICO 2023/2024

TESTO IN USO: Camagni Paolo, Nikolassy Riccardo, Nuovo Tecnologie e progettazione di sistemi informatici e di telecomunicazioni (volume 1), Hoepli

CONTENUTI DISCIPLINARI

Modulo 0 (Attività propedeutiche)

Conoscenza della classe, presentazione dei moduli di teoria e di laboratorio in cui si articola la materia ed indicazioni operative sulle modalità di svolgimento delle verifiche e della loro valutazione

Breve storia dei sistemi di calcolo: l'era meccanica (dalla pascalina al progetto Analytical Engine di Babbage fino ai primi calcolatori meccanici a schede perforate), l'era elettromeccanica (dai primi computer programmabili Z1 e Z3 a relay all'IBM Harvard Mark I programmabile con nastro di carta perforato) e l'era elettronica con la prima generazione di computer (dall'invenzione del triodo all'ENIAC e all'EDVAC, il Whirlwind funzionante in "tempo reale" e l'uso delle memorie a nuclei magnetici) e la seconda generazione di computer dell'era elettronica (dalla scoperta dell'effetto transistor ai circuiti integrati ai microprocessori e sviluppo della famiglia di microprocessori Intel X86)

Breve storia del Personal-Computer e del sistema operativo MS-Windows

Modulo 1 (Introduzione alla rappresentazione delle informazioni e la codifica dei numeri)

La rappresentazione delle informazioni

Cos'è un computer, differenza tra informazioni e dati e tra codifica e decodifica

Significato di sistema di comunicazione (messaggio, trasmettitore, ricevitore, codice, canale e protocollo) e di comunicazione a livelli (trasmissione di significato, di simboli e di segnali fisici) e relativi esempi

Significato di alfabeto sorgente (insieme di valori da rappresentare) e di alfabeto in codice (insieme di simboli per rappresentare l'alfabeto sorgente)

Significato di codifiche a lunghezza fissa (ed esempi guidati-dialogati di calcolo del numero minimo di simboli necessari per codificare un insieme di valori) e di codifiche a lunghezza variabile

Concetti di ridondanza e di codice ridondante

Esempi di codifiche a lunghezza fissa ed a lunghezza variabile partendo da un alfabeto sorgente e da un alfabeto in codice; ulteriori esempi guidati-dialogati di calcolo del numero minimo di simboli (lunghezza minima delle stringhe) necessari per codificare un determinato insieme di informazioni

Tipologie di informazioni e le rispettive codifiche che verranno trattate durante il corso (i numeri naturali, relativi e reali, i simboli alfanumerici, le immagini ed i suoni)

Differenza tra disturbo e rumore e loro influenza sulla trasmissione di un segnale

Codifiche dei numeri

Rappresentazione dei numeri naturali: il numero come entità astratta ed il numerale come sua rappresentazioni (usando una codifica con un sistema di numerazione), sistemi di numerazione additivi/sottrattivi, sistemi posizionali per rappresentare i numeri come sequenze (stringhe) di cifre (coefficienti delle potenze della base) e relativi esempi (i sistemi unario, binario, ternario, quinario, ottale, a base 12 ed esadecimale)

Procedimenti per convertire in base 10 un numero rappresentato in una base qualsiasi: il metodo "delle potenze" (applicazione della definizione di sistema posizionale) ed il metodo di Horner (algoritmo efficiente per il calcolo di un polinomio di grado n che utilizza solo n moltiplicazioni e n addizioni) nella variante per la conversione della parte intera e nella variante per la conversione della parte frazionaria) e relativi esercizi guidati-dialogati

Procedimenti per convertire in una base qualsiasi un numero rappresentato in base 10: il metodo delle "divisioni successive" (per la conversione della parte intera) ed il metodo delle "moltiplicazioni successive" (per la conversione della parte frazionaria) e relativi esercizi guidati-dialogati

Conversioni di base di un numero da una base qualsiasi ad un'altra base qualsiasi, sia passando per la sua rappresentazione nella base 10 sia mediante un metodo semplificato nel caso in cui la base di partenza e quella di arrivo siano una la potenza dell'altra, ed esercizi guidati-dialogati di conversione tra le basi "binarie" (2, 4, 8 e 16)

Le quattro operazioni aritmetiche fondamentali tra numeri binari "puri" (naturali): la somma col metodo "del riporto", la sottrazione "diretta" col metodo "del prestito" (e determinazione sia del massimo numero rappresentabile sia di quanti numeri sono rappresentabili con un certo numero di cifre), il problema dell'overflow aritmetico (l'eventuale impossibilità di rappresentare il risultato di un'operazione con le cifre a disposizione) ed il relativo *flag-bit* usato negli elaboratori per rilevare quest'evento); la moltiplicazione (ed il caso particolare della moltiplicazione per una potenza della base), la divisione (ed il caso particolare della divisione per una potenza della base) e rispettivi esercizi guidati-dialogati

Significato di complemento alla base "diminuita" e di complemento alla base e la loro applicazione al sistema binario con le operazioni rispettivamente di complemento "a uno" e di complemento "a due"

Rappresentazioni dei numeri binari interi (con segno): la rappresentazione in "modulo e segno" e la sua inadeguatezza per rappresentare i numeri interi negativi (sia per il problema nell'effettuare le operazioni aritmetiche, a causa del diverso verso di percorrenza della retta dei numeri tra interi positivi ed interi negativi, che per il problema della doppia rappresentazione dello zero), la rappresentazione in "complemento a uno" dei numeri interi negativi (che risolve il problema del diverso verso di percorrenza della retta dei numeri tra interi positivi e interi negativi ma non risolve il problema della doppia rappresentazione dello zero, ed il suo range di rappresentazione), la rappresentazione in "complemento a due" dei numeri interi negativi (che risolve anche il problema della doppia rappresentazione dello zero ed il suo range di rappresentazione) e relativi esempi guidati-dialogati (anche dei rispettivi range di rappresentazione)

La somma algebrica di numeri in complemento a 2 (anche per effettuare sottrazioni, mediante la trasformazione del sottraendo nel corrispondente numero negativo), la relativa tecnica per intercettare l'eventuale overflow aritmetico (l'impossibilità di rappresentare il risultato di un'operazione con le cifre a disposizione) confrontando nella colonna delle cifre più significative il riporto entrante con quello uscente (e ripasso dell'altra tecnica già trattata per intercettare l'overflow aritmetico con operazioni aritmetiche tra numeri in binario puro), motivo per cui l'overflow aritmetico non può avvenire con addendi aventi segno discorde e relativi esercizi guidati-dialogati

Il metodo dell'eccesso (polarizzazione) 2^{n-1} per rappresentare numeri interi: suo significato e sua utilità per non dover avere un simbolo per rappresentare il segno di un numero intero, suo range di rappresentazione e relativi esempi guidati-dialogati con le regole pratiche partendo dalla rappresentazione decimale del numero, sommandogli la polarizzazione (il valore 2^{n-1}) e convertendo il risultato ottenuto in binario, oppure partendo dalla rappresentazione in complemento a 2 del numero e complementandone il bit più significativo

Differenza tra la rappresentazione dei numeri reali in virgola fissa (e relative problematiche) e la loro rappresentazione in virgola mobile (rappresentazione "scientifica" di un numero mediante segno, mantissa ed esponente e scelta della forma "normalizzata" della mantissa al fine di poterne "sottindendere" la parte intera)

La codifica binaria in virgola mobile nel formato standard IEEE-P754: forma normalizzata della mantissa (avente la parte intera con il solo bit 1 "nascosto") e codifica polarizzata dell'esponente (in eccesso $2^{n-1}-1$), i diversi formati (precisione singola, doppia, doppia estesa e quadrupla) e relativi esercizi guidati-dialogati per la conversione di un numero reale dalla base 10 (posizionale) alla base 2 in IEEE-P754 in precisione singola (rappresentata in base esadecimale) e viceversa

Intervallo di numeri rappresentabili con mantissa normalizzata ed il concetto di underflow aritmetico (impossibilità di rappresentare il risultato di un'operazione se il suo valore assoluto è minore del valore assoluto del numero più piccolo rappresentabile in forma normalizzata), uso degli esponenti "riservati" 0 e 255 (rispettivamente per mantissa denormalizzata e per casi speciali), arrotondamento e troncamento e relativi esercizi guidati-dialogati

Calcolo dei valori estremi di rappresentazione (del valore assoluto del massimo e del minimo numero rappresentabile) e dell'intervallo dei numeri rappresentabili con mantissa normalizzata con IEEE-P754 in precisione singola (e relativa rappresentazione grafica) e conseguente impossibilità a rappresentare numeri il cui valore assoluto è minore di tale minimo

Rappresentazioni particolari mediante l'uso della mantissa denormalizzata e degli esponenti "riservati" (esponente con tutti i bit a zero, per rappresentare lo zero ed i numeri con valore assoluto minore del valore assoluto del numero più piccolo rappresentabile, oppure esponente con tutti i bit a 1, per rappresentare infinito e NaN) e relativi esercizi guidati-dialogati

Arrotondamento e troncamento e relativi esercizi guidati-dialogati

Modulo 2 (Circuiti logici e sistema di elaborazione)

Algebra booleana

Introduzione all'algebra booleana: cenni storici sulla sua nascita come logica "proposizionale" da parte di George Boole e sulla sua applicazione all'elettronica da parte di Claude Shannon

Costanti e variabili booleane, gli operatori fondamentali AND, OR e NOT, gli operatori derivati (universali) NAND e NOR e l'operatore EXOR (ed esempi del suo utilizzo come comparatore, complementatore pilotato e generatore di parità), esempi del loro funzionamento (tramite proposizioni e per analogia con circuiti elettrici) e le rispettive rappresentazioni algebriche, tabelle della verità, descrizioni funzionali e porte logiche

Regole di precedenza degli operatori booleani, proprietà di base e proprietà particolari (teoremi) dell'algebra booleana, semplificazione guidata-dialogata di semplici espressioni logiche mediante l'utilizzo di tali proprietà e disegno dei corrispondenti circuiti logici

Circuiti logici

Introduzione ai circuiti logici: concetto di circuito logico, uso delle porte logiche per disegnare circuiti logici che rappresentano espressioni booleane e cenni all'utilizzo dei transistor per la realizzazione delle porte logiche

Costruzione di circuiti logici contenenti solo porte NAND oppure solo porte NOR (universali) equivalenti alle porte logiche fondamentali AND, OR e NOT (e dimostrazione di tale equivalenza mediante

semplificazione guidata-dialogata delle relative espressioni booleane usando le proprietà dell'algebra booleana)

Le due tipologie di circuiti logici (combinatori e sequenziali) e rispettive caratteristiche e differenze

Analisi dello schema logico e del funzionamento di alcuni circuiti logici combinatori notevoli, facendo anche riferimento al concetto di indirizzo ed alla differenza tra collegamenti in serie ed in parallelo: interruttore di controllo, maschera, decoder (e costruzione guidata-dialogata di un decoder con 3 bit di input di controllo), multiplexer (selettore) e demultiplexer (e costruzione di quest'ultimo a partire da un decoder), comparatore di due word di n bit (usando una porta EXOR per ognuna delle n coppie di bit delle due word ed una porta OR)

Esempio di collegamento di un multiplexer con un demultiplexer per selezionare e distribuire diversi flussi di dati (comunicazioni tra coppie di soggetti) facendoli transitare su un unico collegamento condiviso

Utilità dei decoder nell'indirizzare le locazioni della memoria centrale e per indirizzare diversi dispositivi di input-output, facendo particolare riferimento ai concetti di indirizzo, di system-bus (e funzionamento dei suoi componenti address-bus, data-bus e control-bus all'interno di un sistema di elaborazione), di collegamento in parallelo e di collegamento in serie

Analisi dello schema e del funzionamento dei circuiti combinatori aritmetici half-adder (semisommatore) e full-adder (sommatore completo) e loro utilizzo in parallelo per costruire un full-adder (sommatore parallelo) a n bit composto da una serie di sommatori completi, ognuno per sommare coppie di bit di pari posizione di word di qualsiasi dimensione (anche i bit meno significativi al posto di un half-adder, usando il carry-in per incrementare di 1 un dato in input)

Esempio di utilizzo del sommatore completo per realizzare un circuito aritmetico-logico (ALU) ad 1 bit, che effettua operazioni logiche (AND, OR e NOT) o somme aritmetiche (con riporto) tra due bit in input, in base ad un input di controllo di due bit (che viene decodificato da un decoder, portando solo uno dei quattro risultati in output); utilizzo di n ALU a 1 bit in parallelo per realizzare un'ALU a n bit, facendo particolare riferimento all'input di enable, che ne attiva il funzionamento (ad es. sincronizzandolo con un segnale di clock), e ad un complementatore pilotato, che complementa uno dei due bit di input

I circuiti logici sequenziali: caratteristiche generali, schema logico, funzionamento e tabella delle transizioni dei circuiti bistabili per la memorizzazione di un bit latch S-R, latch D e latch J-K, ciascuno sia di tipo asincrono che di tipo sincrono col clock, ed il loro utilizzo per realizzare memorie volatili SRAM (e cenni alle loro differenze rispetto alle memorie DRAM realizzate con condensatori)

La fase di "trasparenza" dei latch sincronizzati (il periodo in cui il clock è alto durante il quale è possibile alterare la memorizzazione col passaggio di segnali), i problemi (oscillazioni del segnale in input) che potrebbero avvenire durante questa fase dei latch (il cui cambiamento di stato è controllato dal livello alto/basso del clock) e la soluzione a questi problemi mediante l'uso dei circuiti flip-flop (il cui cambiamento di stato è invece controllato dal fronte di salita/discesa del clock)

I flip-flop di tipo master-slave (a memoria ausiliaria), realizzati con due latch collegati in serie ed attivati da segnali di abilitazione complementari, ed i flip-flop di tipo edge-triggered, realizzati con un latch al cui ingresso di abilitazione (clock) è collegato un circuito rilevatore di transizioni (CRT), che genera brevissimi impulsi in corrispondenza del fronte di salita del clock (ed analisi dello schema logico e del funzionamento di questo semplice circuito logico), e le rispettive differenze tra questi due tipi di flip-flop

Rappresentazioni grafiche dei latch (attivati dal clock alto) e dei flip-flop (attivati dal fronte di salita del clock) e delle rispettive varianti a "logica negativa" (latch attivati dal clock basso e flip-flop attivati dal fronte di discesa del clock); eventuale presenza in questi circuiti degli ingressi supplementari asincroni Clear (CLR) e Preset (PR), rispettivamente per "forzare" la scrittura di uno 0 oppure un 1 indipendentemente dal clock, e le loro rappresentazioni grafiche

Esempio di schema logico e funzionamento di un circuito logico che realizza un registro (insieme di flip-flop di tipo D collegati tra loro) sincronizzato col clock, per memorizzare valori binari di 8 bit

Schema logico e funzionamento di un circuito logico che realizza una semplice memoria indirizzabile composta da 4 registri, ciascuno contenente 3 flip-flop di tipo D (per memorizzare 4 word, ciascuna di 3 bit), introducendo i concetti di address-bus (su cui arriva un "indirizzo" di 2 bit necessario per attivare tramite un decoder solo uno dei 4 registri), di control-bus (per comandare la lettura oppure la scrittura del/nel registro, attivato tramite la decodifica dell'indirizzo) e di data-bus (su cui arrivano i tre bit da

memorizzare nel registro attivato in scrittura oppure su cui vengono portati i tre bit letti dal registro attivato in lettura)

Uso combinato di flip-flop sincroni per realizzare registri (a scorrimento): analisi dello schema e del funzionamento di registri che memorizzano dati binari arrivati in sequenza ordinata (seriali) e li restituiscono in parallelo (Serial-Input Parallel-Output o SIPO) oppure nella stessa sequenza in cui sono arrivati (Serial-Input Serial-Output o SISO) e di registri che memorizzano dati binari arrivati in parallelo e li restituiscono in parallelo (Parallel-Input Parallel-Output o PIPO) oppure in sequenza ordinata (Parallel-Input Serial-Output o PISO)

Introduzione al simulatore di circuiti online Tinkercad-Circuits: interfaccia grafica e componenti elettronici disponibili sul simulatore e creazione di un primo circuito su breadboard che realizza un latch S-R mediante un integrato 7402 (con quattro porte NOR), due interruttori (set e reset), un LED ed una resistenza

Struttura interna e funzionamento del microprocessore (CPU)

Introduzione all'architettura ed al funzionamento del calcolatore: il modello di calcolatore di Von Neumann, i suoi tre sottosistemi (microprocessore, memoria centrale e dispositivi di I/O) ed il loro collegamento mediante system-bus (composto da address-bus, data-bus e control-bus)

Il linguaggio macchina e la struttura (formato binario) di istruzione macchina (binaria) e di sua decodifica (nell'Instruction-Decoder), esempio di formato a 16 bit delle istruzioni macchina del microprocessore didattico usato (codifica del gruppo di istruzioni a cui un'istruzione appartiene, dell'eventuale modo di indirizzamento usato, del codice operativo dell'istruzione e degli eventuali registri usati oppure dell'offset)

Concetto di ISA (Instruction Set Architecture), eventuale compatibilità di diverse architetture di microprocessori a livello ISA (e relativi esempi) e differenze tra microprocessori di tipo RISC e di tipo CISC

Significato di linguaggio mnemonico assembly, cenni sulla sua nascita, suo rapporto col linguaggio macchina (corrispondenza quasi univoca tra un'istruzione assembly ed un'istruzione macchina) e ripasso della differenza tra linguaggi di basso livello (assembly e linguaggio macchina) e linguaggi di alto livello (logici)

L'assemblatore, la sua differenza rispetto al compilatore ed il significato della compilazione nella scrittura del software

Significato di "ciclo macchina" (ciclo fetch-decode-execute), le sue quattro fasi con cui viene eseguita una singola istruzione macchina (instruction-fetch, instruction-decode, eventuale operand-fetch ed execute) ed esempio di funzionamento passo-passo della fase di instruction-fetch (prelievo dell'istruzione macchina dalla memoria centrale all'indirizzo indicato nel registro Program-Counter e sua memorizzazione nel registro Instruction-Register, in cui essa viene decodificata)

Presentazione di un modello didattico (semplificato) di microprocessore RISC a 16 bit composto da unità di controllo (CU), unità aritmetico-logica (ALU) e registri interni (ciascuno da 16 bit), differenza tra registri special-purpose (dedicati) e registri generici (general-purpose) e concetti di "parallelismo" di un elaboratore e di macchina load-and-store

Memorizzazione dei dati nelle celle della memoria centrale con la tecnica little-endian (suo significato e differenza rispetto alla tecnica big-endian), invio da parte della CU di un indirizzo sull'address-bus per accedere (tramite sua decodifica) ad una cella della memoria centrale oppure ad un dispositivo di I/O, funzioni della CU (guida del sistema di elaborazione in base alla decodifica di ogni istruzione macchina), dell'ALU (esecuzione di calcoli ed aggiornamento dei flag di stato per registrare eventi che si verificano col susseguirsi delle varie istruzioni durante l'esecuzione di un programma) e del control-bus (con i suoi tre "fili" di controllo R/W, M/IO e W/B)

Funzionamento passo-passo della fase di instruction-fetch nel microprocessore didattico RISC (prelievo dell'istruzione macchina dalla memoria centrale all'indirizzo indicato nel Program-Counter e sua memorizzazione nell'Instruction-Register, dove verrà decodificata nella fase successiva di instruction-decode)

Concetto di modo di indirizzamento (modalità di espressione dell'indirizzo di un operando nell'accesso alla memoria centrale) ed i tre modi di indirizzamento previsti dal microprocessore didattico per trovare un

operando nella memoria centrale (operando immediato, indirizzo assoluto ed indirizzo in registro)

Funzionamento dell'operand-fetch per prelevare un operando dalla memoria centrale e portarlo in un registro generico (memory-load), rispettivamente con i modi di indirizzamento operando immediato, indirizzo assoluto ed indirizzo in registro (facendo particolare riferimento al formato della rispettiva istruzione macchina), e la loro corrispondenza alle modalità d'uso dei dati in un linguaggio di alto livello (C)

Funzionamento dell'operand-fetch per prelevare un operando da un registro generico e portarlo nella memoria centrale (memory-store) con i due modi d'indirizzamento indirizzo assoluto ed indirizzo in registro (e motivo dell'impossibilità del modo di indirizzamento operando immediato con memory-store) e la loro corrispondenza alle modalità d'uso dei dati in un linguaggio di alto livello (C)

Il linguaggio assembly del microprocessore didattico RISC: i quattro gruppi di istruzioni disponibili, analisi del gruppo di istruzioni load-and-store e del gruppo di istruzioni aritmetico-logiche e rispettivi esempi (facendo particolare riferimento alla codifica binaria delle corrispondenti istruzioni macchina, ai tre modi di indirizzamento trattati, all'opcode ed ai registri da usare)

Esempio di un semplice programma in C (che somma i contenuti di due variabili intere e di un valore costante e salva il risultato in una terza variabile intera) e sua riscrittura in assembly usando le direttive per riservare celle della memoria centrale per i dati, corrispondenti alla dichiarazione delle variabili in C (e anche cenni a come vengono gestiti i vettori a basso livello): il caricamento di un valore costante in un registro con operando immediato (prelevandolo dalla memoria centrale subito dopo l'istruzione), il caricamento di dati dalla memoria centrale in registri con indirizzo assoluto e l'uso dell'istruzione ADD per sommare questi dati presenti nei registri (facendo anche particolare riferimento al posizionamento in memoria delle relative istruzioni macchina e dei dati)

Analisi del gruppo di istruzioni di input-output per la lettura o la scrittura di una word o di un byte da/verso un dispositivo di I/O (facendo particolare riferimento alla codifica binaria delle corrispondenti istruzioni macchina, all'opcode ed ai registri da usare), il problema del sincronismo tra la velocità di elaborazione della CPU e la velocità di trasferimento dei dati con i dispositivi di I/O e cenni ad una sua soluzione tramite la tecnica obsoleta del polling (realizzata mediante un ciclo di verifica del completamento dell'operazione di I/O)

Analisi del gruppo di istruzioni di controllo per "saltare" ad un indirizzo di memoria diverso da quello presente nel Program-Counter (modificandone l'indirizzo memorizzato): i salti incondizionati (con indicazione "immediata" in memoria dell'indirizzo a cui saltare oppure con indicazione "diretta" nell'istruzione dell'offset da sommare algebricamente all'indirizzo presente nel Program-Counter) ed i salti condizionati (dipendenti dai singoli valori presenti nei diversi flag del registro di stato ed il loro utilizzo per alterare l'esecuzione di un programma in base agli eventi che si verificano col susseguirsi delle varie istruzioni eseguite dalla CPU)

Le istruzioni di controllo CALL e RET per realizzare a basso livello le subroutine dei linguaggi di alto livello (e loro significato come sottoprogrammi) mediante l'uso dell'area di memoria stack (funzionante con logica LIFO) e del registro dedicato Stack-Pointer (contenente l'indirizzo della prima posizione libera "in cima" allo stack): l'istruzione CALL per "chiamare" una subroutine, a cui viene trasferito il controllo dopo aver sospeso l'esecuzione del programma chiamante (copiando in "cima" allo stack l'indirizzo contenuto nel Program-Counter, che viene poi sovrascritto con l'indirizzo della prima istruzione della subroutine, ed incrementando lo Stack-Pointer), e l'istruzione RET per ripristinare l'esecuzione del programma a partire da dove era stato sospeso (copiando nel Program-Counter dalla "cima" dello stack l'indirizzo precedentemente memorizzato e decrementando lo Stack-Pointer)

L'uso dei salti condizionati ed incondizionati per realizzare in assembly i "blocchi" logici della programmazione strutturata (implementati dalle istruzioni dei linguaggi di alto livello) quali selezione semplice (blocco IF) e multipla (blocco IF-ELSE), ripetizione pre-condizionale (blocco WHILE) e post-condizionale (blocco DO-WHILE)

Analisi guidata-dialogata di un semplice programma esemplificativo in assembly che somma tre numeri in memoria (interpretabili come elementi di un vettore di un linguaggio di alto livello, ciascuno memorizzato su 8 bit) mediante un ciclo di somme e l'utilizzo degli indirizzi come dati (che vengono incrementati per poter accedere all'elemento successivo), facendo particolare riferimento al posizionamento in memoria delle relative istruzioni macchina e dei dati

Analisi guidata-dialogata di un programma in assembly e del corrispondente programma in linguaggio

macchina, che carica in memoria centrale (a partire da un indirizzo arbitrario indicato nel programma) singoli caratteri inseriti da tastiera finché questi sono diversi da "invio" (corrispondente al simbolo ASCII 'CR'), con cui viene invece caricato in memoria il carattere 0, che conclude in memoria la sequenza (stringa) dei caratteri inseriti, facendo particolare riferimento all'offset (presente all'interno delle istruzioni di controllo del flusso) che viene sommato algebricamente all'indirizzo presente nel Program-Counter (per alterare il flusso di esecuzione del programma)

Concetto di scope (ambito di visibilità) di una variabile, significato di allocazione delle variabili locali in una subroutine (le ultime allocate devono essere le prime ad essere deallocate) ed il suo funzionamento a basso livello mediante le istruzioni assembly PUSH, per salvare i dati contenuti nei registri prima dell'esecuzione della subroutine (salvando in "cima" allo stack il dato contenuto in un registro), e POP, per ripristinare i dati precedentemente salvati dopo il "ritorno" dalla subroutine (ripristinando il dato nel registro, trasferendolo dalla "cima" dello stack)

Architetture CISC e RISC: caratteristiche e differenze architetturali e funzionali tra CPU di tipo CISC e CPU di tipo RISC e funzionamento delle unità di controllo microprogrammate delle CPU di tipo CISC (aventi istruzioni complesse e di lunghezza variabile) rispetto alla decodifica (diretta) delle istruzioni (semplici e di lunghezza fissa) nelle unità di controllo delle CPU di tipo RISC

La tecnica del pipelining: ripasso del ciclo macchina senza pipeline, significato di pipeline e la logica di funzionamento del ciclo macchina di una CPU con pipeline (esecuzione contemporanea di fasi diverse del ciclo macchina di istruzioni diverse) e la sua realizzazione mediante la suddivisione della relativa unità di controllo in unità funzionali indipendenti (ognuna specializzata per una fase del ciclo macchina); problemi legati all'uso di pipeline (conflitti strutturali, conflitti tra dati e conflitti di controllo) e loro possibili soluzioni; concetto di scalarità di una CPU e cenni alle architetture superscalari (aventi più pipeline funzionanti in parallelo)

La struttura fisica della memoria centrale e la memoria cache

Introduzione alla struttura fisica della memoria centrale organizzata come matrice di "celle" elementari (ciascuna contenente un bit) e del loro raggruppamento in "locazioni" (blocchi di n celle elementari), a cui si accede con R linee di riga (collegate ad un decoder di riga per la selezione di una sola riga di celle) e C linee di colonna (collegate ad un decoder di colonna per la selezione di una sola colonna di locazioni di memoria), che insieme costituiscono un address-bus di R+C linee d'indirizzo

La memoria centrale: serie di array di celle elementari (ciascuna per memorizzare 1 bit) per memorizzare in forma binaria i programmi in esecuzione, concetto di capacità di una memoria, suddivisione della memoria centrale in una memoria di tipo non "volatile" (impiego, relativi tipi ROM, PROM, EPROM, EEPROM e flash e rispettive tecnologie impiegate e limiti) ed in una memoria di tipo "volatile" RAM (motivi della scelta dell'acronimo, impiego, relativi tipi SRAM e DRAM e rispettive tecnologie impiegate e limiti)

La struttura fisica della memoria centrale organizzata come matrice di "celle" elementari e del loro raggruppamento in "locazioni" (blocchi di N celle elementari), a cui si accede con R linee di riga (collegate ad un decoder di riga per la selezione di una sola riga di celle) e C linee di colonna (collegate ad un decoder di colonna per la selezione di una sola colonna di locazioni di memoria), che insieme costituiscono R+C linee d'indirizzo

Semplici esercizi guidati-dialogati per il calcolo del numero di linee di indirizzo dell'address-bus necessarie per indirizzare celle elementari di una memoria di dimensione nota, sia singolarmente sia a gruppi di 4, 8 o 16 celle (possibili dimensioni di ciascuna locazione di memoria)

La memoria cache: nascita, scopo, organizzazione (livelli della memoria cache ed evoluzione del loro posizionamento, dalla motherboard alla CPU) e funzionamento (in caso di hit ed in caso di miss)

Prestazione dei sistemi di elaborazione

Concetto di potenza di calcolo, problematicità del confronto tra prestazioni di CPU aventi architetture differenti, fattori che influenzano le prestazioni di una CPU (insieme delle istruzioni, dimensione e numero dei registri, scalarità e frequenza di clock) e calcolo guidato-dialogato della stima del tempo di esecuzione da parte di una CPU di un programma avente un numero di istruzioni noto (conoscendo il numero medio di cicli di clock necessari per ogni istruzione macchina e la durata di un ciclo di clock, cioè il reciproco della sua frequenza)

Fattori che influenzano le prestazioni complessive di un sistema di elaborazione, difficoltà di confronto delle prestazioni di diverse architetture (dato che le prestazioni complessive dipendono dal tipo di programma eseguito) ed i principali indici prestazionali (MIPS, MFLOPS e SPECmarks) per confrontare le prestazioni di diversi sistemi di elaborazione, il loro significato ed i rispettivi limiti

Modulo 3 (I codici digitali)

Codici digitali pesati

Le codifiche binarie alfanumeriche: cenni alle codifiche alfanumeriche usate in passato (e relative problematiche), la codifica a lunghezza fissa ASCII standard e le sue proprietà (primi 32 "simboli" di controllo, sequenzialità dei "simboli" per le cifre decimali e per le lettere e distanza fissa di 32 bit tra lettere maiuscole e lettere minuscole) e le codifiche ASCII estese (aventi diverse codepage)

La codifica Unicode (ed il concetto di codepoint a 32 bit), la sottocodifica a lunghezza variabile UTF-8 (ed esempi guidati-dialogati del suo utilizzo) e cenni alle sottocodifiche UTF-16 e UTF-32

Uso di codici binari a lunghezza fissa per rappresentare cifre decimali (per evitare ambiguità d'interpretazione legate all'uso del codice binario "puro" di lunghezza variabile)

Il codice a lunghezza fissa BCD (o 8-4-2-1): sue caratteristiche e differenze rispetto al binario "puro", codifiche utilizzabili e motivi del suo uso, corrispondenza tra le cifre in BCD ed i corrispondenti simboli in ASCII, le operazioni aritmetiche di somma e sottrazione in BCD, differenza tra unpacked e packed BCD e la codifica del segno

Il codice a lunghezza fissa Aiken (o 2-4-2-1): sue caratteristiche e differenze rispetto al codice BCD, codifiche ammesse, la sua proprietà dell'autocomplementarità (e ripasso del concetto di complemento alla base e di complemento alla base diminuita e relativi esempi in base 10 ed in base 2) e motivi del suo uso

Il codice a lunghezza fissa "quinario" (o 5-4-2-1), sue caratteristiche, codifiche ammesse e motivi del suo uso

I codici a lunghezza fissa "biquinario" (o 5-0 e 4-3-2-1-0) e "2 su 5" (o 6-3-2-1-0): loro caratteristiche, codifiche ammesse, la loro proprietà di autoverifica con conteggio fisso (a rivelazione d'errore), motivi del loro uso ed esempio di applicazione del codice "2 su 5" con codici a barre

Codici digitali non pesati

I codici digitali non pesati: assenza di pesi per la stringa di codifica e la loro capacità di rappresentare non solo cifre numeriche

Il codice a lunghezza fissa "eccesso 3": sue caratteristiche, codifiche ammesse, motivi del suo uso, la sua proprietà dell'autocomplementarità e l'operazione aritmetica della somma in "eccesso 3"

Il codice "riflesso" a lunghezza variabile Gray: sue caratteristiche, sua costruzione per "riflessione", la sua proprietà della "ciclicità", motivi del suo uso e sua derivazione a partire da un numero binario puro (mediante le operazioni di shift a destra e somma)

Il codice a lunghezza fissa "eccesso 3 riflesso": sue caratteristiche, la sua proprietà della "ciclicità" e sua derivazione a partire da un numero binario "in eccesso 3" (mediante le operazioni di shift a destra e di somma)

Ulteriori codici digitali e loro caratteristiche e proprietà: il codice "uno su N", il codice "a sette segmenti" (per i display a sette segmenti) ed il codice a matrice di punti (per i display LCD)

Codici per la rilevazione e la correzione degli errori

Introduzione ai codici per la rilevazione e la correzione degli errori: generalità sulla rappresentazione fisica di un bit e sui fenomeni che possono alterarne il valore (e cenni al "rumore" elettromagnetico causato dal fenomeno dell'induzione elettromagnetica), necessità di bit aggiuntivi per rilevare eventuali

errori ed eventualmente correggerli (ridondanza della codifica) e differenza tra codici rilevatori e codici correttori di errore

Significato di distanza di Hamming e sua rappresentazione mediante il reticolo di Hamming (e relativi esempi fino a 4 bit); distanza minima di Hamming e concetto di codice rilevatore di ordine n per poter rilevare n errori (e necessità di usare codifiche con distanza di Hamming tra i dati di almeno $n + 1$), relativo esempio guidato-dialogato ed analisi del suo significato usando il reticolo di Hamming

Il concetto di "parità" (e di "disparità") ed il controllo di parità "con ridondanza singola" come codice rilevatore di errore (e ripasso della proprietà della porta logica EXOR con n ingressi per realizzarlo): suoi limiti ed esempi del suo funzionamento (con codeword di 7 bit più 1 bit di ridondanza per la parità) e del suo impiego (nelle memorie dei computer)

Il metodo polinomiale CRC (Cyclic-Redundancy-Check) come codice rilevatore di errore: concetto di "checksum" e suo calcolo come resto della "sottrazione modulo due" (con EXOR bit a bit) di un dato binario con i coefficienti binari di un polinomio "generatore" ed esempi guidati-dialogati del suo funzionamento

Significato di codice correttore di ordine n per poter correggere fino ad n errori

Il metodo LRC (a parità "incrociata") come codice correttore di errore che, oltre ai bit di parità calcolati per ogni word "trasversalmente" (con "ridondanza singola"), aggiunge anche un'ulteriore word di parità (checksum) formata dai bit di parità calcolati "longitudinalmente" al blocco di n word, esempi guidati-dialogati del suo funzionamento ed analisi del motivo del suo fallimento se gli errori sono più di uno

Il metodo di Hamming come codice correttore di errore ed esempi guidati-dialogati del suo funzionamento con dati di 7 e di 8 bit, sia in trasmissione (per generare codeword di 11 e di 12 bit) che in ricezione (per trovare e correggere eventuali errori nelle codeword ricevute)

Codifiche delle immagini

Introduzione alla codifica digitale delle immagini raster e vettoriali: concetti di immagine raster (scalare) e di immagine vettoriale e loro differenza di funzionamento

Significato di digitalizzazione come procedura composta da una fase di campionamento e da una fase di quantizzazione (usando n bit) ed il loro rispettivo significato

La codifica digitale delle immagini raster mediante digitalizzazione (codifica) dei loro pixel (campionamento "spaziale") e quantizzazione con n bit ("profondità" di codifica) ed esempi del suo funzionamento assegnando ad ogni pixel 1 bit ("bianco e nero"), 2 bit, 4 bit oppure 8 bit

La codifica delle immagini raster a colori, che usa almeno tre colori di base per codificare ogni colore (una "profondità" di colore di n bit per ognuno dei colori di base) ed i modelli cromatici RGB (additivo) e CYMK (sottrattivo); differenza tra alcune codifiche bitmap notevoli per profondità di colore

Significato di palette di colori e suo uso per codificare con la tecnica raster solo un sottoinsieme dei colori utilizzabili (tecnica a colori indicizzati o color-map) e suoi vantaggi rispetto alla codifica raster true-color

La trasparenza e suo uso come trasparenza semplice (come un colore di una palette di un'immagine indicizzata) oppure come trasparenza "per strato alpha" o alpha-channel (aggiungendo un byte ad ogni pixel di un'immagine raster in modo da avere più livelli di trasparenza) e concetto di alpha-blending

Concetto di definizione di un'immagine raster come sua "dimensione" (in numero di pixel) e calcolo del suo "peso" (in byte) come prodotto della definizione per la profondità di colore

Concetto di risoluzione grafica di un dispositivo ("densità" di pixel di un dispositivo), sua differenza rispetto alla definizione di un'immagine e le unità di misura della risoluzione come numero di pixel per pollice (PPI) per i dispositivi video o come punti tipografici per pollice (DPI) per i dispositivi di stampa ed esempio di calcolo di PPI/DPI di un dispositivo col teorema di Pitagora (partendo dalla risoluzione del dispositivo e dalla sua diagonale in pollici); influenza della risoluzione sulla grandezza di un'immagine (inversamente proporzionali tra loro)

Le immagini vettoriali: caratteristiche, loro costruzione mediante funzioni matematiche ed il rendering (processo di visualizzazione di un'immagine vettoriale mediante un dispositivo con una data risoluzione)

Codifiche per le informazioni multimediali

Introduzione alla multimedialità (animazioni e suoni): concetto di multimedialità, realizzazione di animazioni con sequenze di immagini fisse (fotogrammi o frame) e motivo fisiologico del suo funzionamento (persistenza delle immagini sulla retina); cenni al funzionamento del vecchio tubo catodico ed alla scansione delle immagini di tipo interlacciato o progressivo

La multimedialità: cenni storici sulla sua nascita, concetto di framerate, i formati video più diffusi (AVI, MPEG e MOV) ed esercizi guidati-dialogati di calcolo delle dimensioni in byte di file multimediali

Realizzazione di suoni digitali mediante trasformazione di un suono (onda acustica) in un segnale analogico (mediante un trasduttore) e successivi campionamento nel tempo (concetto di bitrate e ripasso del concetto di frequenza) e quantizzazione (su n livelli di quantizzazione dipendenti dalla profondità di codifica); i formati audio più diffusi (WAV e MP3) e relativi esercizi guidati-dialogati

Il protocollo audio MIDI e la sua caratteristica di codificare una nota ed il modo di riprodurla (anziché digitalizzare un suono) e cenni sullo standard General MIDI

La compressione dei dati

La compressione dei dati: significato di "compressione" e differenza tra i metodi di compressione senza (lossless) e con (lossy) perdita di informazione

La compressione lossless, casi di sua applicabilità (compressori ed alcuni formati grafici) ed il formato GIF

La compressione lossy, casi di sua applicabilità, il formato JPEG e cenni alle quattro fasi del suo funzionamento

La codifica di Huffman (algoritmo di compressione statistica): significato di compressione statistica, funzionamento dell'algoritmo di Huffman, che effettua una compressione statistica per ottenere una codifica a lunghezza variabile efficiente (ridotta) per caratteri alfanumerici rispetto ad una loro codifica a lunghezza fissa (costruzione di un albero binario in base alla frequenza con cui compaiono i caratteri alfanumerici in un testo e suo utilizzo per l'assegnazione di una codifica binaria ad ogni carattere), e relativi esercizi guidati-dialogati

Modulo 4 (Il sistema operativo)

Generalità sui sistemi operativi

Introduzione ai sistemi operativi: il sistema operativo come software di base "gestore" e "protettore" delle risorse della macchina fisica; cenni sull'evoluzione storica dei sistemi operativi dai primi sistemi "a lotti" (batch) per il caricamento automatico e l'elaborazione in sequenza dei programmi, ai sistemi operativi "multiprogrammati" (con più programmi coesistenti in memoria), agli odierni sistemi multiprogrammati in time-sharing

Le fasi del processo di "avvio" ("bootstrap") del sistema operativo dopo il test dei componenti hardware (POST): caricamento da un indirizzo convenzionale di una memoria flash della prima istruzione di salto incondizionato al codice del BIOS (firmware che prepara la macchina per il caricamento del sistema operativo dalla memoria di massa alla memoria centrale)

Le quattro funzioni principali di un sistema operativo (che verranno trattate durante il modulo): la gestione dei processi, la gestione della memoria centrale, la gestione della memoria di massa (il file-system) e la gestione dei sistemi di I/O ed il modello a strati con cui tali funzioni vengono rappresentate (skin-onion-model); i concetti di kernel e di shell (a riga di comando oppure con interfaccia grafica) e le rispettive funzionalità (ed il concetto di macchina virtuale), la gestione delle risorse condivise e la necessità della loro protezione

Funzionamento dei sistemi operativi "a lotti" (batch) per il caricamento automatico e l'elaborazione in sequenza dei programmi, dei sistemi operativi "multiprogrammati" (con più programmi in esecuzione coesistenti in memoria) e degli odierni sistemi multiprogrammati in time-sharing

Gestione delle risorse hardware e software condivise mediante una macchina "virtuale" (simulata) con cui un utente interagisce e relativi esempi (modifica di file, coda di stampa e gestione della memoria centrale)

La gestione dei processi

La gestione dei processi per la loro assegnazione alla CPU: differenza tra programma e processo, il descrittore di processo (PCB) e le informazioni in esso contenute (e concetto di "contesto" di un processo)

I processi di tipo CPU-bound e I/O-bound ed introduzione al funzionamento delle relative tecnologie usate per realizzarli (interrupt, per la gestione delle interruzioni di un processo e per salvarne lo stato di esecuzione, e DMA, per limitare l'intervento della CPU nelle operazioni di I/O)

Diagramma a stati dei possibili stati di un processo di un sistema operativo multiprogrammato in time-sharing e le transizioni tra essi, significato delle code di attesa dei processi (waiting-list e ready-list)

Il significato di "descrittore" di un processo" (PCB) e le informazioni in esso contenute

Il concetto di "contesto" di un processo ed il cambiamento di "contesto" (context-switch) quando la CPU viene assegnata ad un altro processo

I modi di funzionamento della CPU (user-mode e kernel-mode) ed il loro utilizzo nel funzionamento del cambiamento di "contesto" (context-switch) quando la CPU viene assegnata ad un altro processo

Significato di cheduling dei processi, obiettivi perseguiti da un sistema operativo mediante lo scheduling dei processi (con relativi esempi per sistemi batch, interattivi e real-time) e significato di processo di tipo pre-emptive (con prerilascio) e non pre-emptive (senza prerilascio)

Analisi di alcune logiche di scheduling per sistemi multitasking: l'algoritmo First-Come-First-Served (senza prerilascio), l'algoritmo Shortest-Job-First (senza prerilascio) e la sua variante Shortest-Remaining-Time-First (con prerilascio, che tiene conto del tempo residuo di esecuzione del processo), l'algoritmo con "priorità" (sia senza prerilascio che con prerilascio) e la sua variante per evitare il fenomeno della starvation dei processi con bassa priorità (mediante l'uso di priorità variabili in base all'ageing dei processi in coda), l'algoritmo Round-Robin ("a rotazione", variante dell'algoritmo First-Come-First-Served ma con prerilascio periodico del processo al termine della sua time-slice per l'uso del time-sharing) e l'algoritmo Multiple-Level-Feedback-Queues (MLFQ) e rispettivi esempi guidati-dialogati di calcolo del tempo medio di attesa

Cenni alle differenze tra le logiche di scheduling adottate dai sistemi Windows, Linux e Macintosh

Cenni ai processi indipendenti e cooperanti ed alle problematiche della loro sincronizzazione

La gestione della memoria centrale

La gestione della memoria centrale: significato di spazio di indirizzamento virtuale e spazio di indirizzamento reale per sistemi multiprogrammati

Concetto di rilocazione degli indirizzi (traduzione degli indirizzi di memoria logici generati dal processo in indirizzi fisici immessi sull'address-bus) e differenza tra la rilocazione statica e la rilocazione dinamica, quest'ultima realizzata mediante l'uso del dispositivo Memory-Management-Unit (MMU) per la conversione "al volo" (binding) degli indirizzi logici generati da un processo (usati in fase di programmazione di basso livello o in fase di compilazione del programma di alto livello) in indirizzi fisici (in cui sono effettivamente memorizzati i segmenti o le pagine dei processi caricati nella memoria centrale) quando occorre inviarli sull'address bus

Le tecniche di allocazione contigua della memoria: schema a partizioni fisse oppure schema a partizioni variabili e le rispettive problematiche di frammentazione interna ed esterna

Le tecniche di allocazione non contigua della memoria mediante la memoria virtuale (memoria utilizzabile superiore a quella fisicamente disponibile tramite l'uso del page-file nella memoria di massa come "estensione" della memoria centrale utilizzabile dalla CPU oltre la sua dimensione fisica realmente

disponibile): segmentazione dei processi con segment-table, paginazione dei processi con page-table e segmentazione dei processi con paginazione dei segmenti, rispettivi vantaggi e svantaggi e rispettivi esempi di funzionamento (e la protezione della memoria centrale da accessi non consentiti mediante bit di protezione)

Il trattamento del page-fault (eccezione quando la pagina richiesta non è presente nella page-table), il principio di località spazio-temporale per la scelta della pagina logica "vittima" (da rimuovere dalla memoria centrale per liberarne il frame occupato) ed i criteri di scelta (resident-set-management oppure con algoritmo di sostituzione della pagina da rimuovere e sua logica "ottimizzata") e l'uso del dirty-bit nel caso in cui la pagina da rimuovere sia stata modificata

Algoritmi di sostituzione per la scelta della pagina "vittima" da rimuovere dalla memoria centrale: First-In-First-Out, Least-Recently-Used e la sua variante Not-Recently-Used ("clock-algorithm", sia nella versione che usa solo il bit d'uso che nella versione che usa anche il dirty-bit oltre al bit d'uso), Least-Frequently-Used e Most-Frequently-Used

Il fenomeno del thrashing (fenomeno di forte rallentamento dell'elaboratore causato dalla degenerazione della memoria virtuale in presenza di troppi processi in rapporto alla quantità disponibile di memoria centrale, i quali causano molteplici page fault, che generano continue letture e scritture del page file e, quindi, limitano l'elaborazione dei processi in esecuzione provocando un forte degrado delle prestazioni del sistema di elaborazione) e l'uso dello swap di alcuni processi nella memoria secondaria (per consentire l'evoluzione degli altri, decongestionando il sistema)

Concetto di resident-set di un processo (necessario affinché questo possa procedere nella sua esecuzione) e la logica del working-set, con cui il sistema operativo decide per ogni processo il numero di pagine da caricare nei frame fisici (alternativa alla demand-paging, con cui le pagine vengono caricate nei frame fisici in base alle richieste causate dai page-fault)

La gestione del file-system

Introduzione al File-System: funzionalità, esposizione della memoria di massa all'utente come macchina simulata che presenta una struttura logica gerarchica di file e di directory (path assoluto e relativo), organizzazione fisica della memoria secondaria (suddivisa in "blocchi", ciascuno contenente la stessa quantità di dati), funzionamento dei dischi ottici e degli HDD, questi ultimi costituiti da più dischi magnetici "impilati" e da più testine "a pettine" che ne leggono i dati memorizzati nei blocchi (e richiami al fenomeno fisico dell'induzione elettromagnetica che ne consente il funzionamento), ciascuno individuato da un disco, da un "cilindro" (composto da "tracce" circolari concentriche su ogni disco) e da un "settore" radiale, e significato di tempo di posizionamento e di tempo di latenza per la lettura dei dati in un blocco

Tipi di File-System ed il rispettivo sistema operativo che li utilizza (e breve excursus storico su FAT e NTFS e sulle relative versioni di DOS e di Windows)

Significato di file (e di directory) e di "descrittore" di un file ed il suo contenuto (memorizzato all'interno della directory in cui si trova il file)

Concetto di "volume" come area di una memoria di massa suddivisa in blocchi fisici di dimensione fissa, che usa un determinato file-system e la sua directory principale (device-directory) e che contiene i descrittori dei file e delle sottodirectory in essa contenuti (e le relative informazioni incluse)

Significato di allocazione di un file su memoria di massa (sua suddivisione in blocchi logici e loro memorizzazione nei blocchi fisici della memoria di massa), il funzionamento dei metodi di allocazione dei file (allocazione contigua, allocazione indicizzata ed allocazione concatenata) e cenni alla deframmentazione di Windows

Le operazioni sui file ed il caricamento in memoria centrale del descrittore del file che viene aperto

La gestione dei dispositivi di I/O

Introduzione alla gestione dei dispositivi di I/O: l'asincronicità delle operazioni della CPU rispetto alle operazioni di I/O (il problema del sincronismo tra l'elevata velocità di elaborazione delle istruzioni della CPU ed i tre tipi di trasferimento dati con i dispositivi di I/O che verranno trattati)

La tecnica (obsoleta) del "trasferimento programmato" da/verso i dispositivi di I/O (polling), implementata mediante un ciclo di verifica del completamento dell'operazione di I/O (realizzata con istruzioni assembly che impostano un flag del registro di stato se l'operazione di I/O è completata)

La tecnica del "trasferimento per interruzione" da/verso i dispositivi di I/O (interrupt), adatta al multitasking dei moderni sistemi operativi (con la quale, al verificarsi di eventi esterni, vengono inviati dei segnali hardware di "interruzione" dai dispositivi di I/O alla CPU che attivano l'esecuzione di subroutine di servizio necessarie per l'esecuzione della specifica operazione di I/O), il mascheramento dell'interrupt (per evitare la ricezione di ulteriori interrupt), le linee di interrupt ad altissima priorità "non mascherabili" (e relativa necessità) e la gestione degli interrupt (tramite linee di richieste indipendenti oppure tramite un controller esterno)

La tecnica del "trasferimento diretto in memoria" da/verso i dispositivi di I/O (DMA), che evita che per ogni unità di dato (word) da trasferire venga generato un interrupt per la CPU (ed attivata la relativa subroutine di servizio), e l'uso del DMA-controller, dispositivo hardware che consente il trasferimento diretto di "blocchi" di dati tra un dispositivo di I/O e la memoria centrale senza coinvolgere la CPU, che avvia solo il trasferimento del "blocco" di dati (che avverrà in autonomia) senza dover essere "interrotta" per ogni unità di dato trasferita (risolvendo quindi il problema della "lentezza percepita" della CPU quando il trasferimento singolo dei dati avviene tramite interrupt)

Modulo 5 (Il sistema operativo Linux)

Introduzione ai sistemi operativi ed a Linux, concetto di open-source, differenza tra interfaccia utente di tipo GUI e di tipo CLI ed introduzione alla shell Bash

Introduzione ai comandi Bash Linux: la struttura di base di un comando Bash (e relativi opzioni ed argomenti), la cartella root (radice /), l'esecuzione dei comandi ls, ls -a, ls -la, pwd, cd, touch, mkdir, il significato dei caratteri jolly [.] e [..] e l'uso del comando man per visualizzare il manuale relativo ad un comando

La struttura dei comandi, le opzioni e gli argomenti e relativi esempi guidati-dialogati

Comandi esterni e comandi interni, spiegazione dei comandi help, info e man

Il file-system di Linux e la sua struttura, differenza tra percorso relativo e percorso assoluto, percorsi relativi speciali ed il comando tree

Analisi del comando mv ed esempio per rinominare o spostare un file e rinominare o spostare una cartella

Uso del comando mkdir per creare una cartella

Uso del comando rmdir per eliminare una cartella vuota ed rm -r * per eliminare una cartella al cui interno sono presenti file

Il comando cp per copiare file

Vedere i permessi di file e cartelle in lettura, scrittura ed esecuzione mediante la combinazione "rwx" (e la sua interpretazione in codice binario) e l'uso del comando chmod per assegnare o cambiare tali i permessi sia tramite la notazione in ottale, sia con l'utilizzo dei caratteri

Il comando gzip per comprimere un file e gzip -l per visualizzare i dettagli del file compresso, il comando gunzip per decomprimere un file, il comando tar -cf NOME_ARCHIVIO [lista file] per inserire più file in un file "archivio" ed il comando tar -czf NOME_ARCHIVIO per decomprimere i file da un file "archivio" (ed uso dei caratteri '*' e '?' per comprimere più file)

Esecuzione di un file .sh ed introduzione agli script in Bash: utilizzo di "#!/bin/sh" per indicare l'interprete da utilizzare per eseguire lo script

Utilizzo delle variabili dichiarandole col prefisso \$ prima del nome della variabile (\$NOME_VARIABILE) ed utilizzo di pwd come comando per eseguire la stampa a video del percorso assoluto della directory corrente

Realizzazione di uno script in Bash con un ciclo FOR che stampa le informazioni (stringhe) presenti in una variabile e un ciclo FOR che stampa i nomi di file e cartelle presenti nella working-directory; introduzione al ciclo WHILE

Vittorio Veneto, 05/06/2024

Firma del docente _____

Firma di due studenti _____
