



Verifica scritta valida per l'orale di Sistemi e Reti - Teoria

Data: **12-11-2024**

Nome		Cognome	
-------------	--	----------------	--

Parte 1 - Domande a scelta multipla (10 punti)

Seleziona la risposta corretta per ciascuna domanda (seleziona UNA SOLA)

1. Quale tipo di memoria sfrutta il principio di località temporale?

- A) La ROM
- B) La RAM
- C) La Cache
- D) L'EPROM

2. Quale caratteristica distingue la SRAM dalla DRAM?

- A) È non volatile
- B) Non necessita di refresh
- C) Ha capacità maggiore
- D) È più economica

3. Quale tipo di architettura prevede l'utilizzo di un opcode per le sue operazioni?

- A) Architettura a due operandi
- B) Architettura a tre operandi
- C) MIPS
- D) DIPS

4. Quale periferica si occupa di gestire le interruzioni all'interno di un modulo condiviso?

- A) Module controller
- B) Controller
- C) PCI/e
- D) DMA

5. Quale tipo di memoria ROM è elettricamente riprogrammabile?

- A) EPROM
- B) EEPROM
- C) PROM
- D) ESPROM

6. L'Instruction set di tipo CISC:

- A) Ha poche istruzioni semplici
- B) Ha molte istruzioni complesse
- C) È tipico dei microcontrollori
- D) È sempre più veloce del RISC

7. L'architettura SIMD:

- A) Esegue istruzioni diverse su dati diversi
- B) Esegue la stessa istruzione su dati diversi
- C) Non può essere implementata in hardware
- D) È più lenta di una architettura sequenziale

8. La cosiddetta "legge di Amdahl" implementa il principio di "località" dei dati: cosa vuol dire?

- A) I dati sono già disponibili in memoria, non occorre riprenderli tramite fetch/decode/execute
- B) I dati si trovano vicini, quindi non dobbiamo spostarci; usiamo dei bus per prenderli
- C) Non è un concetto applicabile alle CPU
- D) Diminuisce il parallelismo; non si usa

9. La SRAM rispetto alla DRAM:

- A) È più lenta ma più economica
- B) È più veloce ma più costosa
- C) Ha maggiore capacità
- D) Richiede refresh periodico

10. Quali parti compongono il linguaggio Assembly (per come l'abbiamo visto noi: x86)?

- A) Deassembler/Unlinker
- B) Disassembler/Reassembler
- C) Assembler/Linker
- D) Assembler

Parte 2 – Vero o Falso (10 punti)

Indica se le seguenti affermazioni sono Vere (V) o False (F) – l'opzione di scelta è sulla colonna a dx

- | | |
|--|-------|
| 1. La memoria DRAM è più costosa della SRAM. | (V/F) |
| 2. L'istruzione MOV tra due locazioni di memoria è consentita in tutte le architetture x86. | (V/F) |
| 3. Per effettuare un indirizzamento, le componenti sono SOLO due: base address ed offset. | (V/F) |
| 4. L'istruzione JMP in Assembly corrisponde ad un IF in programmazione. | (V/F) |
| 5. La sintassi delle istruzioni è "Opcode Destinazione Sorgente" | (V/F) |
| 6. Il ciclo macchina è composto da 5 fasi: IF, ID, EX, MEM e WB | (V/F) |
| 7. Il DMA (Direct Memory Access) permette il trasferimento diretto di dati tra memoria e periferiche senza coinvolgere la CPU. | (V/F) |
| 8. La pipeline permette di eseguire più fasi di istruzioni diverse contemporaneamente. | (V/F) |
| 9. Nell'architettura MIMD ogni processore esegue la stessa istruzione su dati diversi. | (V/F) |
| 10. La cache prevede due tipi di caratteristiche per trovare i dati: hit e miss | (V/F) |

Parte 3 – Domande aperte e pratiche (20 punti)

Rispondi alle seguenti domande in modo chiaro e preciso (se lo ritieni opportuno, si disegni pure schema e/o figura di interesse, anche dietro i fogli – i dettagli e gli esempi sono apprezzati e contribuiscono a più punti)

1. Descrivi la tecnica della pipeline, spiegando come migliora le prestazioni del processore e quali problemi può presentare. Fai un esempio pratico di come funziona.

Risposta:

2. Spiega le differenze tra architetture CISC e RISC, evidenziando vantaggi e svantaggi di ciascuna. Perché molti processori moderni usano un approccio ibrido?

Risposta:

[illegible]

3. Descrivi i diversi tipi di metodi di indirizzamento nelle architetture x86, spiegando come funzionano e facendo esempi con istruzioni assembly.

Risposta:

[illegible]

4. Data la seguente sequenza di istruzioni assembly x86 (sintassi: istruzione – sorgente – destinazione):

```
MOV AX, 5      ; AX = 5
MOV BX, 3      ; BX = 3
MOV CX, 0      ; CX = 0
start:
ADD AX, BX     ; AX = AX + BX
DEC CX         ; CX = CX - 1
CMP CX, -5     ; confronta CX con -5
JNE start
; salta a 'start' se CX ≠ -5
```

- Spiega cosa fa il programma
- Calcola il valore finale di AX
- Indica quante volte viene eseguito il ciclo

Risposta:

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and extend across the width of the page. There are no margins or other markings on the paper.