

Cognome e nome dello studente:

Matricola:

1. [7] Data la CPU di pagina 4, specificare il contenuto di **tutti** i bus, quando è in esecuzione il seguente segmento di codice [5]:

```
0x00000400 or $s5, $t2, $t1
0x00000404 sw $s1, 8($s0)
0x00000408 add $t4, $s5, $s1
0x0000040C addi $t1, $t4, 100
0x00000410 lw $s2, 32($s0)
```

quando l'istruzione di `or` si trova in fase di WB. Specificare sullo schema (con colore o con tratto grosso) quali linee, all'interno dei diversi stadi, trasportino dati utili all'esecuzione dell'istruzione [2]. Ci sono hazard nel codice precedente? Motivare la risposta [1].

2. [5] Cosa sono gli interrupt e le eccezioni? Come vengono gestite dai sistemi operativi? Specificare gli elementi della CPU che sono dedicati alla gestione delle eccezioni e supportano il sistema operativo nel MIPS. Modificare la CPU sopra per potere gestire un'eccezione di "Overflow" e un'eccezione di "Memory Miss". Cosa si intende per mascheramento degli interrupt? Viene praticato nei MIPS?

3. [5] Descrivere come funzionano le seguenti tecniche e dire se sono tecniche principalmente software o hardware e perchè. In alcuni casi la risposta corretta può essere entrambi gli approcci. Identificare quali sono i **punti forti** ed i **punti deboli**.

- a) Superpipeline
- b) Predizione dei salti
- c) Branch prediction buffer
- d) Speculazione
- e) Parallelizzazione dell'esecuzione
- f) Parallelizzazione a livello di parola
- g) Parallelismo implicito ed esplicito
- h) Pipeline superscalari
- i) Pipeline dotate di VLIW
- j) Esecuzione fuori ordine
- k) Reservation station
- l) Buffer di riordino
- m) Ridenominazione dei registri
- n) Branch delay slot
- o) Issue
- p) Hazard
- q) Bolla
- r) Stallo
- s) Cluter
- t) Multi-core
- u) MIPS
- v) MFLOS

4. [6] Disegnare una memoria cache (parte dati + TAG + bit di validità, porte di lettura e scrittura) per un'architettura MIPS a 64 bit, a 4 vie di 4 KByte per banco, e linee di 32 parole (per ciascun banco). Definire cosa rappresenta il campo TAG e dimensionarlo opportunamente. Da quanti bit è costituita questa memoria complessivamente? Cosa succede quando si verifica una miss? Da quale dei due banchi viene scaricato il dato? Perché? Cosa sono i codici di rilevamento e correzione degli errori? Come funziona il codice di Hamming? Fare un esempio.

5. [2] Cosa rappresenta il “roof model”? Cosa rappresenta l’intensità aritmetica? Si riferisce ad una CPU o ad un particolare programma? Quali sono i passi per ottimizzare le prestazioni del codice suggeriti dal roof-model? Cos’è un kernel benchmark? Cos’è lo SPEC?

6. [5] Le memorie sono costituite da una gerarchia costruita secondo criteri ben precisi. Rispondere a queste domande motivando la risposta:

- a) Cosa si intende per gerarchia delle memorie?
- b) Cosa si intende per coerenza e consistenza di una memoria?
- c) A quale tipo di memoria si applicano?
- d) Quali sono i meccanismi messi in atto per garantire la coerenza e la consistenza della memoria nelle architetture mono-processore e nelle architetture multi-processore?
- e) Quali sono i vantaggi e svantaggi di ciascun meccanismo?
- f) Cosa si intende per hit e miss e come vengono gestiti? Chi li gestisce?
- g) Perché le miss sono critiche?
- h) Che differenza c’è tra una miss e un page fault? Cos’è un page fault?
- i) Cos’è la memoria virtuale?
- j) Cos’è la Tabella delle pagine? Dove si trova?
- k) Cos’è il “Translation Lookaside buffer”? Dove si trova?
- l) A cosa servono la memoria virtuale, il TLB e la tabella delle pagine?
- m) Che relazione c’è tra la memoria virtuale e la memoria fisica?
- n) Chi utilizza la memoria virtuale? Chi utilizza la memoria fisica?
- o) Cosa succede quando la CPU chiede una parola alla memoria?

7. [3] Come vengono gestiti gli I/O da un’architettura? Come funziona il “daisy chain”? Cosa si intende per arbitraggio?

8. [3] Spiegare in cosa consiste una CPU multi-ciclo e il principio su cui è basata. Spiegare perché questo progetto è stato abbandonato in favore della pipeline.

### Registri del register file

0	zero constant 0	16	s0 callee saves
1	at reserved for assembler	...	(caller can clobber)
2	v0 expression evaluation &	23	s7
3	v1 function results	24	t8 temporary (cont'd)
4	a0 arguments	25	t9
5	a1	26	k0 reserved for OS kernel
6	a2	27	k1
7	a3	28	gp Pointer to global area
8	t0 temporary: caller saves	29	sp Stack pointer
...	(callee can clobber)	30	fp frame pointer (s8)
15	t7	31	ra Return Address (HW)



