Cognome e nome dello studente: ***Matricola:***

1. [7] Data la CPU di pagina 4, specificare il contenuto di dei registri (parte master e parte slave) quando è in esecuzione il seguente segmento di codice [5]:

```
0x00000400 andi $s5, $t2, 128
0x00000404 lw $s1, 16($s0)
0x00000408 sub $t4, $s5, $s4
0x0000040C addi $t1, $s1, 48
0x00000410 sw $t1, 16($s1)
0x00000414 sub $s2, $s0, $s2
```

quando l'istruzione di `andi` 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]. Si verificano hazard nell'esecuzione del codice precedente? Motivare la risposta [1].

2. [6] 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 "Miss della memoria". Cosa si intende per mascheramento degli interrupt? Viene praticato nei MIPS?

3. [6] 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 predicion 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

4. [8] Disegnare una memoria cache (parte dati + TAG + bit di validità) per un'architettura MIPS a 64 bit, a 4 vie di 4KByte per banco, e linee di 8 parole (per ciascun banco). Definire cosa rappresenta il campo TAG e dimensionarlo opportunamente. Supponiamo che all'inizio i bit di validità siano tutti a 0. Definire cosa succede in corrsipondenza di questo frammento di codice (se si verifica una miss, una hit e dove vengono scritti / letti i dati della cache, quale indirizzo e quale tag vengono associati ad ogni istruzione):

```
sw $t0, 48($zero)
lw $t0, 128($zero)
lw $t0, 112($zero)
sw $t0, 112($zero)
sw $t0, 0($zero)
lw $t0, 48($zero)
lw $t0, 128($zero)
lw $t0, 256($zero)
```

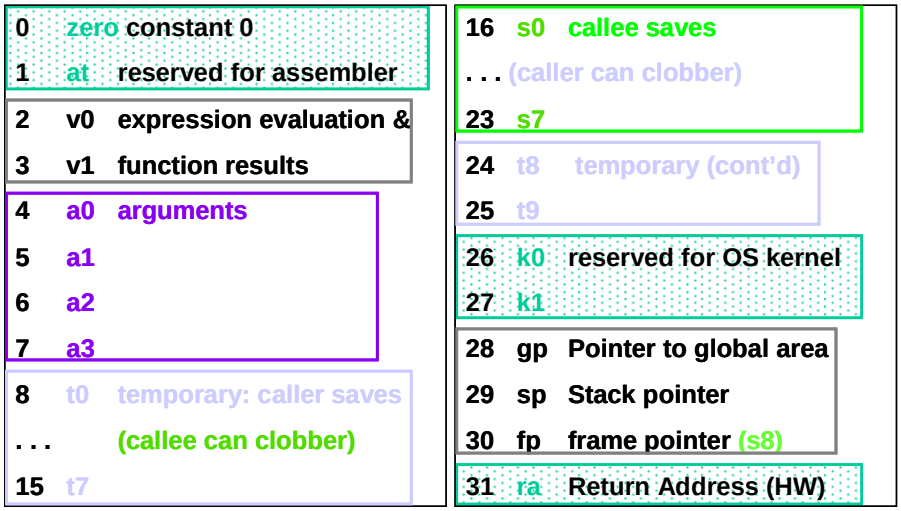
```
lw $t0, 2048($zero)
lw $t0, 4096($zero)
lw $t0, 2072($zero)
```

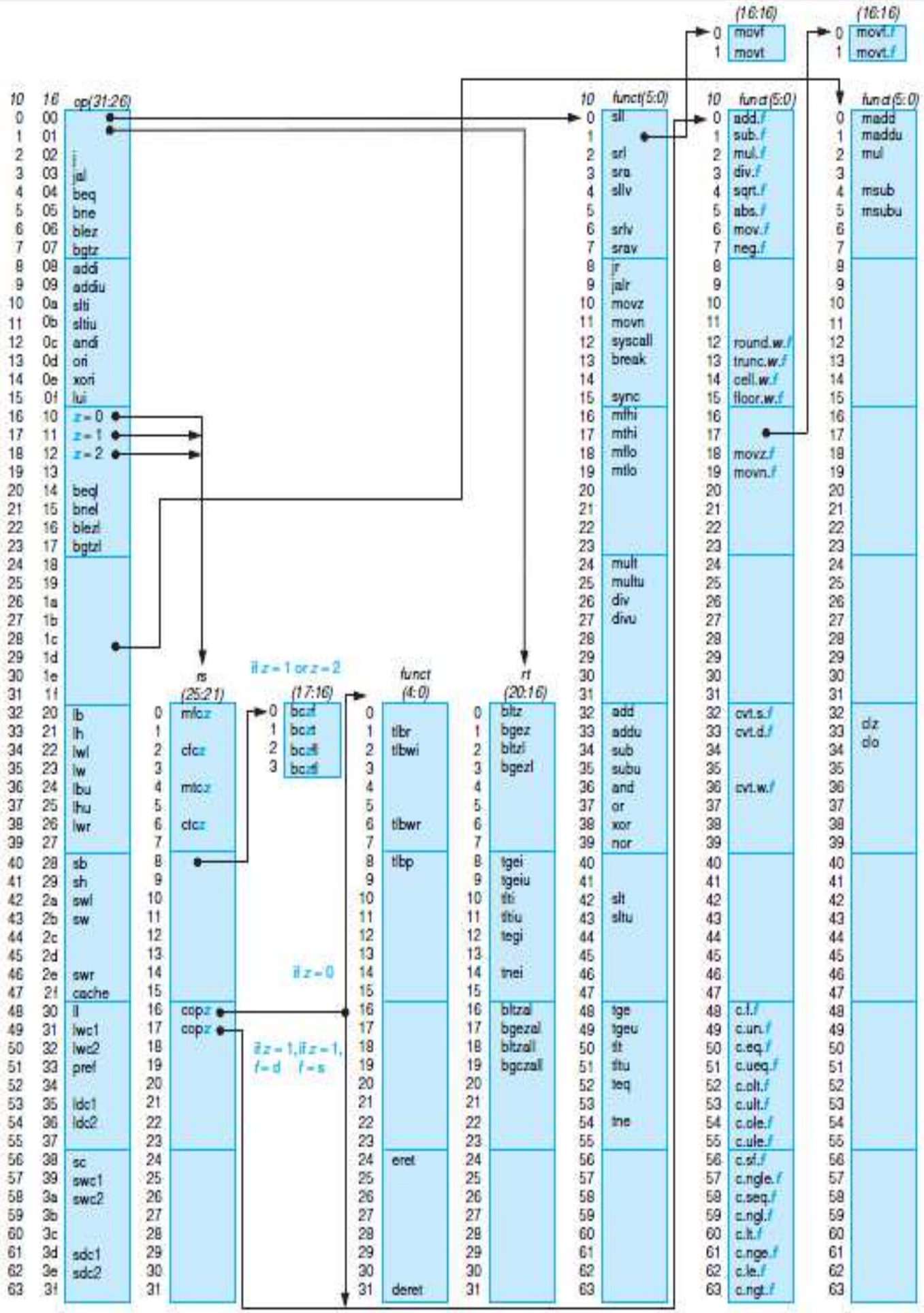
Da quanti bit è costituita questa memoria complessivamente? Cosa sono i codici di rilevamento e correzione degli errori? Come funziona il codice di Hamming?

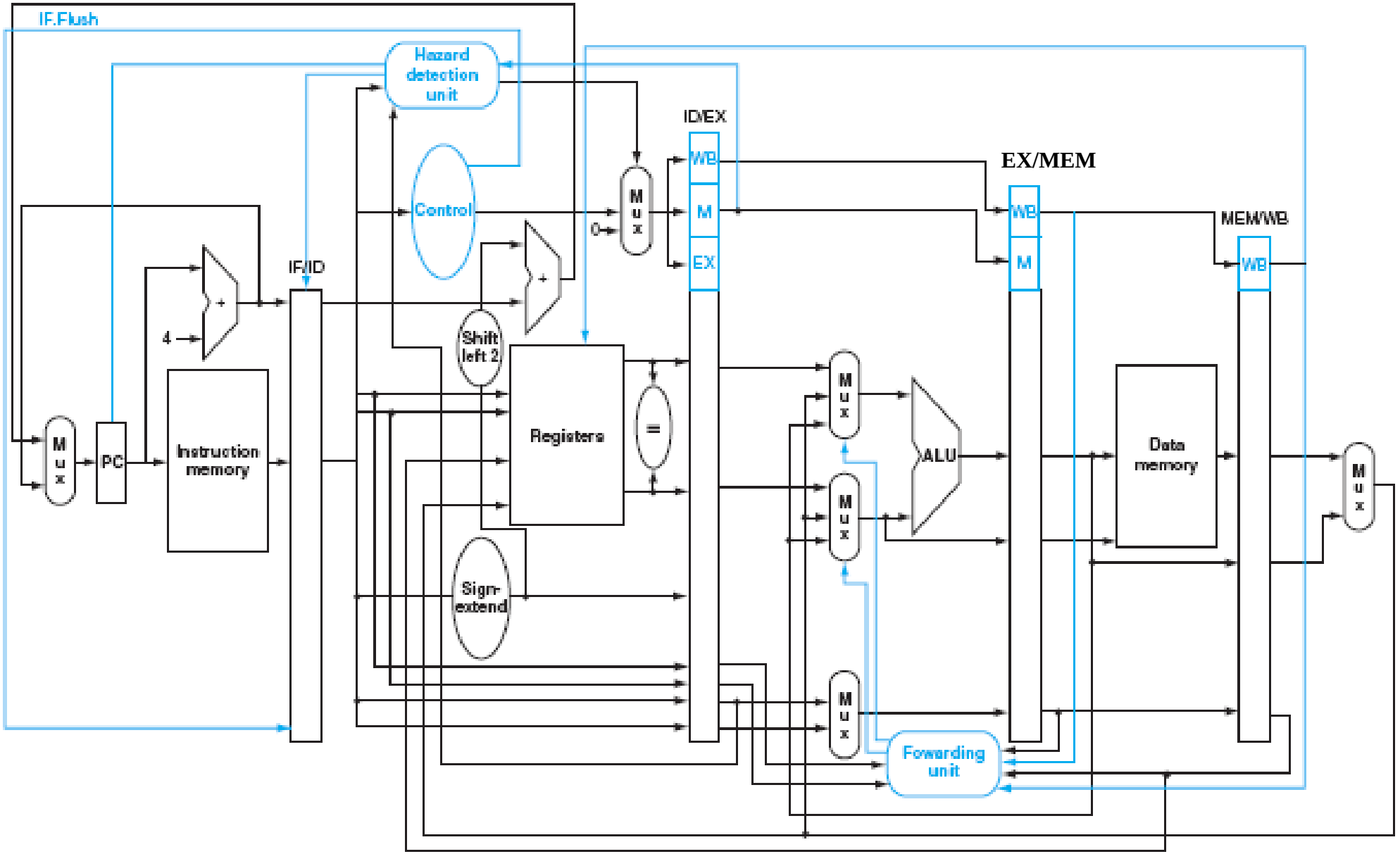
5. [3] Cos'è un cluster? Cos'è un'architettura multi-core? Quali sono le maggiori problematiche per cluster e architetture multi-core? Cosa rappresenta il "roof model"? Cosa rappresenta l'intensità aritmetica? Si riferisce ad una CPU o ad un particolare programma? Quali sono i passi per ottimizzare le prestazioni del codice suggeriti dal roof-model? Cos'è un kernel benchmark? Cos'è lo SPEC?

6. [7] Cosa si intende per gerarchia delle memorie? Cosa si intende per coerenza e consistenza di una memoria? A quale tipo di memoria si applicano? 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? Quali sono i vantaggi e svantaggi di ciascun meccanismo? Cosa si intende per hit e miss e come vengono gestiti? Chi li gestisce? Perché le miss sono critiche? Che differenza c'è tra una miss e un page fault? Cos'è un page fault? Cos'è la memoria virtuale? Cos'è la Tabella delle pagine? Dove si trova? Cos'è il "Translation Lookaside buffer"? Dove si trova? A cosa servono la memoria virtuale, il TLB e la tabella delle pagine? Che relazione c'è tra la memoria virtuale e la memoria fisica? Chi utilizza la memoria virtuale? Chi utilizza la memoria fisica? Cosa succede quando la CPU chiede una parola alla memoria?

Registri del register file




op(31:26)
rs (25:21)
rt (20:16)
funct (4:0)
funct(5:0)
(16:16)
movf
movt
movf.f
movt.f
bc1f
bc1t
bc1fl
bc1tl
if z = 1 or z = 2
if z = 0
if z = 1, if z = 1,
f = d f = s
sll
srl
sra
sllv
srlv
srav
jr
jalr
movz
movn
syscall
break
sync
mfhi
mthi
mflo
mtlo
mult
multu
div
divu
add
addu
sub
subu
and
or
xor
nor
slt
sltu
tge
tgeu
tlt
tltu
teq
tne
add.f
sub.f
mul.f
div.f
sqrt.f
abs.f
mov.f
neg.f
round.w.f
trunc.w.f
ceil.w.f
floor.w.f
movz.f
movn.f
cvt.s.f
cvt.d.f
cvt.w.f
c.f.f
c.un.f
c.eq.f
c.ueq.f
c.olt.f
c.ult.f
c.ole.f
c.ule.f
c.sf.f
c.ngle.f
c.seq.f
c.ngl.f
c.lt.f
c.nge.f
c.le.f
c.ngt.f
madd
maddu
mul
msub
msubu
clz
clo
tlbr
tlbwi
tlbwr
tlbp
eret
deret
bltz
bgez
bltzl
bgezl
tgei
tgeiu
tlti
tltiu
tegi
tnei
bltzal
bgezal
bltzall
bgezall


IF.Flush
Hazard detection unit
ID/EX
EX/MEM
MEM/WB
Control
WB
M
EX
IF/ID
Shift left 2
Registers
ALU
Instruction memory
PC
Sign-extend
Data memory
Fowarding unit
Mux
4
0
=