

Θέμα 1

Ζητούμενο 1

A) θεωρήστε έναν επεξεργαστή MIPS με την κλασική σωλήνωση 5 σταδίων που διδαχθήκατε στο μάθημα και έναν δεύτερο, όπου τα στάδια EX και MEM έχουν αντιμετωπισθεί (χωρίς καμία άλλη τροποποίηση/επέκταση).

i) Δώστε μια αλληλουχία εντολών όπου στον πρώτο επεξεργαστή απαιτείται stall ενώ στο δεύτερο όχι

lw \$t0, 0(\$0)		F	D	X	M	W		F	D	M	X	W			
add \$t1, \$t0, \$zero			F	D	-	X	M	W			F	D	M	X	W

ii) Δώστε μια αλληλουχία εντολών όπου στο δεύτερο επεξεργαστή απαιτείται stall ενώ στον πρώτο όχι.

iii) Αναφέρετε μια θετική και μια αρνητική συνέπεια της αλλαγής αυτής στη σχεδίαση ως προς τον παραγόμενο κώδικα.

Ζητούμενο 2

B) θεωρήστε το κλασικό datapath του MIPS που διδαχθήκατε στο μάθημα, στο οποίο εκτελείται ένα πρόγραμμα με 40% αριθμητικές εντολές, 25% εντολές μνήμης και 35% εντολές διακλάδωσης. Δίνεται ότι μια εντολή διακλάδωσης στη multi-cycle υλοποίηση απαιτεί 2 κύκλους, ενώ στην pipelined υλοποίηση εισάγει κατά μέσο όρο 4.6 stalls. Δίνεται επίσης ότι η pipelined υλοποίηση απαιτεί 25% μεγαλύτερο κύκλο ρολογιού από τη multi-cycle. Β το μέγιστο ποσοστό εντολών μνήμης που μπορούν να δημιουργούν RAW hazards, ώστε η εκτέλεση του προγράμματος να είναι πιο γρήγορη στην pipelined υλοποίηση.

αριθμητικές εντολές: 40%		multi-cycle		pipelined:
εντολές μνήμης: 25%				ratio pipeline = 1,25 · ratio multi
εντολές διακλάδωσης: 35%		2cc		46 stall

Ζητούμενο 3

Γ) Διαφορετικές εκδόσεις του μεταγλωττιστή (compiler) είναι δυνατό να παράγουν για την ίδια εφαρμογή εκτελέσιμα με διαφορετική επίδοση (χρόνο εκτέλεσης) στον ίδιο κάθε φορά επεξεργαστή. Εξηγήστε συνοπτικά δύο λόγους για τους οποίους συμβαίνει αυτό.

1. Βελτιστοποιήσεις κώδικα: Οι compilers εφαρμόζουν διαφορετικές βελτιστοποιήσεις που μπορεί να επηρεάσουν την απόδοση.
2. Υποστήριξη ειδικών χαρακτηριστικών: Ορισμένοι compilers είναι πιο εξειδικευμένοι στην αξιοποίηση ειδικών χαρακτηριστικών του επεξεργαστή, οδηγώντας σε βελτιωμένη απόδοση.

Θέμα 2

Υποθέστε την κλασική αρχιτεκτονική σωλήνωσης του MIPS αποτελούμενη από τα στάδια IF, ID, EX, MEM, WB. Όλα τα στάδια διαρκούν ένα κύκλο. Κατά τον εντοπισμό μιας εντολής άλματος υπό συνθήκη, ο επεξεργαστής κάνει stall τη σωλήνωση μέχρι την επίλυση, η οποία πραγματοποιείται στο στάδιο EX.

Τέλος, υποθέστε ότι η εγγραφή σε ένα καταχωρητή γίνεται στο πρώτο μισό ενός κύκλου, ενώ η ανάγνωση από τον ίδιο καταχωρητή πραγματοποιείται στο δεύτερο μισό του κύκλου. Δίνεται ότι η αρχική τιμή του καταχωρητή \$t9 (ακριβώς πριν το loop) είναι ίση με 0x300 καθώς και το ακόλουθο κομμάτι κώδικα:

LOOP:

```
LW $t4, 8($t2)
ADD $t2, $t4, $t6
SW $t2, 8($t4)
ADD $t5, $t2, $t3
LW $t6, 0($t5)
SW $t6, 4($t2)
ADD $t6, $t6, $t5
ADDI $t9, $t9, -4
BNEZ $t9, LOOP
```

$$\$t9 = 0x300_{16} = 3 \cdot 16^2 \quad \# \text{ loops} = 3 \cdot \frac{16^2}{4} = 32^* = 192$$

Ζητούμενο 1

$$\$t9 = -4$$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
LW	F	D	X	M	W																		
add		F	D	-	-	X	M	W															
sw			F	-	-	D	-	-	X	M	W												
add				F	-	-	D	X	M	W													
LW					F	D	-	-	X	M	W												
sw						F	-	-	D	-	-	X	M	W									
add							F	-	-	D	X	M	W										
addi								F	D	X	M	W											
bnez									F	D	-	-	X	M	W								
LW													F	D									

~total cycles: 21.191 + 23 = 4034 cc

Ζητούμενο 2

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
LW	F	D	X	M	W									
add		F	D	-	X	M	W							
sw			F	-	D	X	M	W						
add				F	D	X	M	W						
LW					F	D	X	M	W					
sw						F	D	X	M	W				
add							F	D	X	M	W			
addi								F	D	X	M	W		
bnez									F	D	X	M	W	
LW												F	D	

~total cycles: 12.191 + 14 = 2.306 cc

Ζητούμενο 3

LOOP:

```

LW $t4, 8($t2)
ADD $t2, $t4, $t6
SW $t2, 8($t4)
ADD $t5, $t2, $t3
LW $t6, 0($t5)
SW $t6, 4($t2)
ADD $t6, $t6, $t5
ADDI $t9, $t9, -4
BNEZ $t9, LOOP

```

LOOP:

```

LW $t4, 8($t2)
ADDI $t9, $t9, -4
ADD $t2, $t4, $t6
SW $t2, 8($t4)
ADD $t5, $t2, $t3
LW $t6, 0($t5)
SW $t6, 4($t2)
ADD $t6, $t6, $t5
BNEZ $t9, LOOP

```

	1	2	3	4	5	6	7	8	9	10	11	12	13
lw	F	D	X	M	W								
addi		F	D	X	M	W							
add			F	D	X	M	W						
sw				F	D	X	M	W					
add					F	D	X	M	W				
lw						F	D	X	M	W			
sw							F	D	X	M	W		
add								F	D	X	M	W	
bnez									F	D	X	M	W
lw											F	D	

~total cycles: 11*191+13=2.114 cc

Θέμα 3

Οι πίνακες περιέχουν στοιχεία κινητής υποδιαστολής διπλής ακρίβειας, μεγέθους 8 bytes το καθένα. Κάνουμε τις εξής υποθέσεις:

- Το πρόγραμμα τελειώνει σε έναν 32-bit επεξεργαστή με ένα μόνο επίπεδο κρυφής μνήμης δεδομένων, η οποία αρχικά είναι άδεια. Η κρυφή μνήμη είναι direct-mapped, write-allocate και έχει χωρητικότητα 256B. Το μέγεθος του block είναι 32 bytes, ενώ η μικρότερη μονάδα δεδομένων που μπορεί να διευθυνσιοδοτηθεί είναι το 1 byte.
- Όλες οι μεταβλητές πλην των στοιχείων των πινάκων αποθηκεύονται σε καταχωρητές του επεξεργαστή κι επομένως οποιαδήποτε αναφορά σε αυτές δεν συνεπάγεται προσπάθεια στην κρυφή μνήμη.
- Σε περίπτωση εντολών assembly οι αναγνώσεις γίνονται με τη σειρά που εμφανίζονται στον κώδικα.
- Οι πινάκες είναι ευθυγραμμισμένοι και αποθηκεύονται κατά γραμμές. Το πρώτο στοιχείο του πίνακα Α βρίσκεται στη διεύθυνση **0x4BABA0** και του πίνακα Β στη **0x5CCECEA0**.

```
int i,j;
double A[32][16], B[512];

for (i=0; i < 32; i++) {
    for (j=0; j < 16; j++)
        A[i][j] = A[i][j] + B[i*16 + j*8];
}
```

Ζητούμενο 1

Α) Δείξτε το σχήμα διευθυνσιοδότησης, δηλαδή πόσα bits χρειάζονται τα πεδία tag, index και block_offset των διευθύνσεων μνήμης του επεξεργαστή.

$\text{αλφάβητο size} = 8 \text{ bytes}$
 $\text{block size} = 32 \text{ bytes}$

$\left. \begin{array}{l} \text{αλφάβητο size} = 8 \text{ bytes} \\ \text{block size} = 32 \text{ bytes} \end{array} \right\} \begin{array}{l} \text{αλφάβητο / block} = 32 / 8 = 4 \\ \text{+ οπρ. κάρτα γραμμής} \end{array} \leadsto \begin{array}{l} A[i][0-3], A[i][4-7], A[i][8-11], A[i][12-15] \\ B[i-6+i3], i \bmod 4 = 0 \end{array}$

cache size = 256 bytes = 2^8 } #blocks = $2^8 / 2^5 = 2^3 = 8 \Rightarrow$ #sets = 8
block size = 32 bytes = 2^5

block size = 2^5 bytes $\Rightarrow$ offset = 5 bits

#sets = $2^3 \Rightarrow$ index = 3 bits

address size = 32 bits $\Rightarrow$ tag = $32 - 3 - 5 = 24$ bits

1° A: $0_x ABABABAO_{16} \rightarrow \dots 1010\ 0000_2$

1° B: $O_x CEECC EA O_w \rightarrow \dots 1010 \ 0000_2$

$\Rightarrow \text{index} = 101_2 = 5$ dan $\text{offset} = 00000_2 = 0$

↳ set 5

Ζητούμενο 2

B) Βρείτε το συνολικό αριθμό hits και misses για όλη την εκτέλεση του παραπάνω κώδικα.

directed map με 8 blocks και 4 στοιχεία / block

$$AC[i][j] \rightarrow \left(\frac{16i+j}{4}\right) \bmod 8 = \left(4i + \frac{j}{4}\right) \bmod 8$$

$$i=2k: AC[i][j] \rightarrow (j/4) \bmod 8$$

$$i=2k+1: AC[i][j] \rightarrow 4 + (j/4) \bmod 8$$

$$BC[16*i+j \bmod 8] \rightarrow \left(\frac{16i+j \bmod 8}{4}\right) \bmod 8 = \left(4i + \frac{j \bmod 8}{4}\right) \bmod 8$$

$$i=2k, j=0-7: AC[i][j] \rightarrow (j \bmod 8 / 4) \bmod 8 = (j/4) \bmod 8$$

$$i=2k, j=8-15: AC[i][j] \rightarrow (j \bmod 8 / 4) \bmod 8 = (j-8/4) \bmod 8 = (j/4) \bmod 8 - 2$$

$$i=2k+1, j=0-7: AC[i][j] \rightarrow 4 + (j/4) \bmod 8$$

$$i=2k+1, j=8-15: AC[i][j] \rightarrow 2 + (j/4) \bmod 8$$

Συνοψως, για ιδιο i, j : $AC[i][j]$ και $BC[16i+j \bmod 8]$: $\begin{cases} j=0-7: \text{idio set} \\ j=8-15: 2 \text{ set διαφορα} \end{cases}$

$$i=0, j=0-15$$

m m m

h m m

h m m

h m m

m m m

h m m

h m m

h m m

m m h

h h h

h h h

h h h

m m h

h h h

h h h

h h h

total: 32 (26 hits 22 misses) = 832 hits και 704 misses

Ζητούμενο 3

Γ) Τι αλλάζει αν η cache είναι write-no-allocate; Υπολογίστε ξανά το συνολικό αριθμό hits και misses για την εκτέλεση του παραπάνω κώδικα.

$$i=0, j=0-7$$

m m m

m m m

m m m

m m m

m m m

m m m

m m m

m m m

$$i=0, j=8-15$$

m h h

h h h

h h h

h h h

m h h

h h h

h h h

h h h

total: 32 (22 hits 26 misses) =
704 hits και 832 misses

Θέμα 4

Ζητούμενο 1

A) Δίνεται το παρακάτω πρόγραμμα σε C καθώς και μια μετάφραση του σε assembly MIPS. Θεωρήστε ότι ο καταχωρητής \$s0 περιέχει τη διεύθυνση του πρώτου στοιχείου του πίνακα array. Συμπληρώστε τα κενά. Σας υπενθυμίζουμε ότι ο καταχωρητής \$0 (ή \$zero) είναι πάντα μηδέν.

```
int array[20];
int *p, N;

p = &array[3];

while (*p != 0) {
    if (*p < 50) *p = *p % N;
    else *p = *p / N;
    p++;
}
```

```

addi $s1, $s0, 12
LOOP: lw $t0, 0($s1)
      beq $t0, $0, END
      div $t0, $s2
      sli $t1, $t0, 50
      beq $t1, $0, ELSE
      mthi $t0
      jmp NEXT
ELSE: mlo $t0
NEXT: sw $t0, 0($s1)
      addi $s1, $s1, 4
      jmp LOOP
END:
```

$s1 \leftarrow \&p[3]$
 $t0 \leftarrow p[3]$
 if $(p[3] == 0)$, end
 $\text{div } t0, N \Rightarrow mthi = t0/N \text{ και } mlo = t0 \bmod N$
 if $!(t0 < 50)$, else
 $t0 = t0 \% N$
 $NENT$
 $t0 = t0 / N$
 $p = t0$
 $*p++$

B) Δίνεται το παρακάτω κομμάτι κώδικα:

```
total = 0;
for(j = 0; j < k; j++) {
    sub_total = 0;
    for(i = 0; i < N; i++) {
        sub_total += A[j*N + i];
    }
    total += sub_total / N;
}
average = total / k;
```

Ζητούμενο 2

Ως αρχιτέκτονες καλείστε να επιλέξετε την κατάλληλη cache για την εκτέλεση αυτού του κώδικα. Για δεδομένη χωρητικότητα και associativity θα επιλέγατε μεγάλο ή μικρό block size; Δικαιολογήστε την απάντησή σας.

τα στοιχεία θα προσπελάσσονται ως εξής:

$j=0: A[0], A[1], A[2], \dots, A[N-1]$

$j=1: A[N], A[N+1], \dots, A[2N-1]$

$j=2: A[2N], \dots$

$\vdots$

$j=k-1: A[(k-1)N], \dots, A[kN-1]$

→ έχουμε υψηλή spatial locality, η οποία είναι πιο αποδοτική για μεγάλο block size