

Θέμα 1

Ζητούμενο 1

i) Για το ίδιο datapath, η διάρκεια του κύκλου ενός single-cycle επεξεργαστή είναι μεγαλύτερη από αυτή ενός multi-cycle επεξεργαστή.

~ Σωστό

single cycle: ο κύκλος είναι ίσος με την πιο αργή ενότητα ($n \times \text{lw}$)

pipelined: ο κύκλος είναι ίσος με το πιο αργό στάδιο ($n \times \text{Memory, ALU}$)

Συνεπώς, $T_{\text{single cycle}} > T_{\text{pipelined}}$

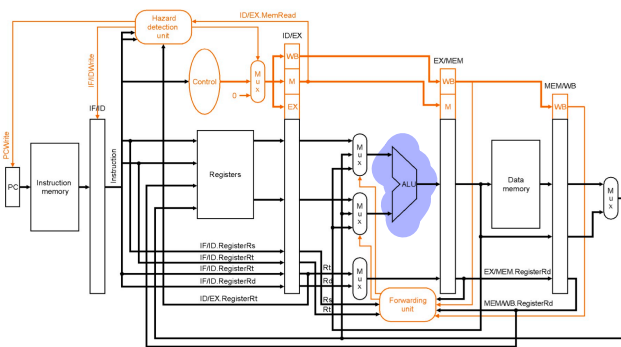
ii) Η αύξηση των σταδίων της σωλήνωσης ενός pipelined επεξεργαστή έχει σαν αποτέλεσμα τη μείωση του CPI.

~ Σωστό

pipeline: $\text{CPI} = 1 + (\text{branches}) \cdot \text{stall}_{\text{branch}} + (\text{lw}) (\text{RAW} \text{ ή } \text{WAR} \text{ ή } \text{WAW} \text{ ή } \text{WAW} \text{ ή } \text{WAW}) \cdot \text{stall}_{\text{data}}$
 αύξηση σταδίων $\Rightarrow$ μείωση stall $\Rightarrow$ μείωση CPI (?)

iii) Σε ένα pipelined datapath χρειάζεται μόνο ένας αθροιστής.

Final Pipeline:



2 αθροιστές:

① $\text{PC} \sim \text{PC} + 4$

② για branches

iv) Κατά την εκτέλεση ενός κώδικα σε ένα pipelined επεξεργαστή, η χρήση καλύτερου compiler δεν επηρεάζει το CPI.

~ Λάθος

καλύτερος compiler μπορεί να οδηγήσει σε μικρότερο CPI, μέσω αναδιατάξης κώδικα ή άλλων τεχνικών

v) Η ενοποίηση 2 σταδίων της σωλήνωσης σε ένα στάδιο, δεν επηρεάζει τη συχνότητα λειτουργίας του επεξεργαστή.

~ Λάθος

η συχνότητα του pipeline καθορίζεται από το πιο αργό στάδιο, το οποίο ενδέχεται να αλλάξει μετά την ενοποίηση

β) Δώστε τους ορισμούς των WAW και WAR εξαρτήσεων. Εξηγήστε ποιες από τις παραπάνω εξαρτήσεις οδηγούν σε κινδύνους και ποιες όχι στην κλασική αρχιτεκτονική σωληνώσης 5 σταδίων του MIPS. Αναφέρετε μερικά παραδείγματα κωδικών που προκαλούν κινδύνους και μερικά που δεν προκαλούν κινδύνους.

- ### Ζητούμενο 3

Εξήγηστε τις έννοιες της χωρικής και χρονικής τοπικότητας των αναφορών στη νήψη, καθώς και οδήγησαν σε κινδύνους και ποιες όχι στην κλασική αρχιτεκτονική των σχολείων.

Ε) Εξηγήστε τις έννοιες της χωρικής και χρονικής τοπικότητας των αναφορών στη νήψη, καθώς και πώς τις εκμεταλλεύονται σε μια κρίση νήψη. Περιγράψτε τα γενικά χαρακτηριστικά των προγραμμάτων που θα παρουσίαζε πολύ υψηλό ποσοστό χρονικής τοπικότητας αλλά πολύ μικρή τοπικότητα ως προς τη νήψη, πολύ υψηλό ποσοστό χωρικής τοπικότητας αλλά πολύ μικρή χρονική τοπικότητα και πολύ υψηλό ποσοστό ενός προγράμματος που θα παρουσίαζε πολύ μικρή χωρική τοπικότητα αλλά πολύ υψηλό ποσοστό τοπικότητας. Για κάθε περίπτωση, δώστε ένα παράδειγμα προγράμματος (με ψευδοκείμενα).

Θέμα 2ο (30%)

- **Temporal Locality:** Στοιχεία που έχουν πρόσφατα προσπελαστεί τείνουν να προσπελαύνονται ξανά στο άμεσο μέλλον
- **Spatial locality:** Γειτονικά στοιχεία όσων έχουν ήδη προσπελαστεί, έχουν αυξημένη πιθανότητα να προσπελαστούν στο άμεσο μέλλον

77

```
for (i=0; i<N; i++)
```

```
for (j=0; j<N; j++)
```

wynn temporary:

for ($j=0$; $j < N$; $j++$)

```
for (i=0; i<N; i++)
```

$$a[i][j] += a[i][j];$$

Θέμα 2

Υποθέστε κλασική αρχιτεκτονική MIPS όπου το στάδιο MEM σπάει σε M1, M2 σε προοπτική μείωσης της διάρκειας κύκλου. Επίλυση εντολών άλματος στο στάδιο EX.
\$t9 = 0x300 και \$t2 = 0x100

LOOP:

```
ADDI $t2, $t2, 4
LW $t1, 100($t2)
LW $t5, 0($t1)
SW $t1, 0($t2)
ADD $t3, $t3, $t5
ADD $t3, $t3, $t4
LW $t4, 0($t2)
ADD $t5, $t4, $t3
ADDI $t9, $t9, -4
BNE $t9, $t2, LOOP
```

Ζητούμενο 1

\$t9 = 3 * 16² και \$t2 = 16² ⇒ #loops = $\frac{2 \cdot 16^2}{2 \cdot 4} = 64$
κάθε loop: \$t9 -= 4, \$t2 += 4

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
addi	F	D	X	M1	M2	W																		
lw		F	D	-	-	-	X	M1	M2	W														
lw			F	-	-	-	O	-	-	-	X	M1	M2	W										
sw							F	-	-	-	O	X	M1	M2	W									
add											F	D	-	-	X	M1	M2	W						
add											F	-	-	O	-	-	-	X	M3	M2	W			
lw															F	-	-	-	D	X	M1	M2	W	
add																		F	D	-	-	-	X	
addi																			F	-	-	-	O	
bne																								F

25 26 27 28 29 30 31 32 33 34 35

add	M1	M2	W																					
addi	X	M1	M2	W																				
bne	D	-	-	-	X	M1	M2	W																
lw					F	D	X	M1	M2	W														

~total cycles: 63 29 + 32

Ζητούμενο 2

	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
addi	F	D	X	M1	M2	W														
lw		F	D	X	M1	M2	W													
lw			F	D	-	-	X	M1	M2	W										
sw				F	-	-	D	X	M1	M2	W									
add							F	D	-	X	M1	M2	W							
add								F	-	D	X	M1	M2	W						
lw									F	D	X	M1	M2	W						
add										F	D	-	-	X	M1	M2	W			
addi											F	-	-	D	X	M1	M2	W		
bne														F	D	X	M1	M2	W	
lw															F	D	X	M1	M2	W

~total cycles: 63: 17 + 20

~total cycles: 63 17 + 20

Zincapelo 3

version 1

version 2

LOOP:

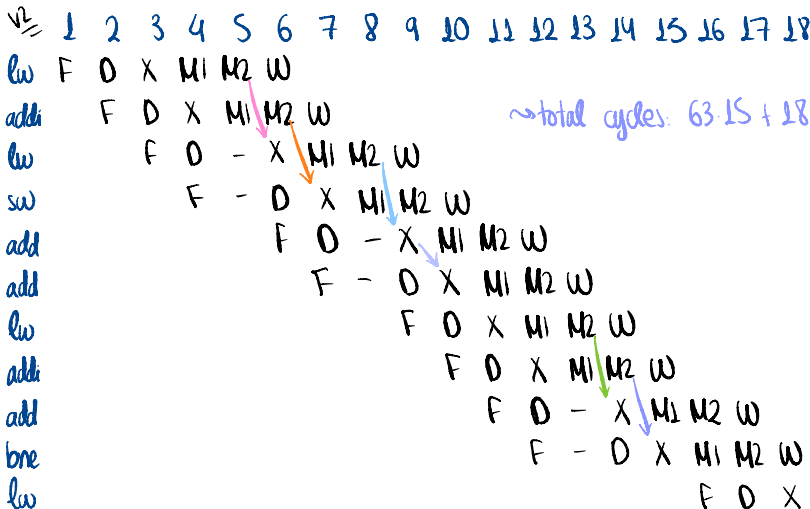
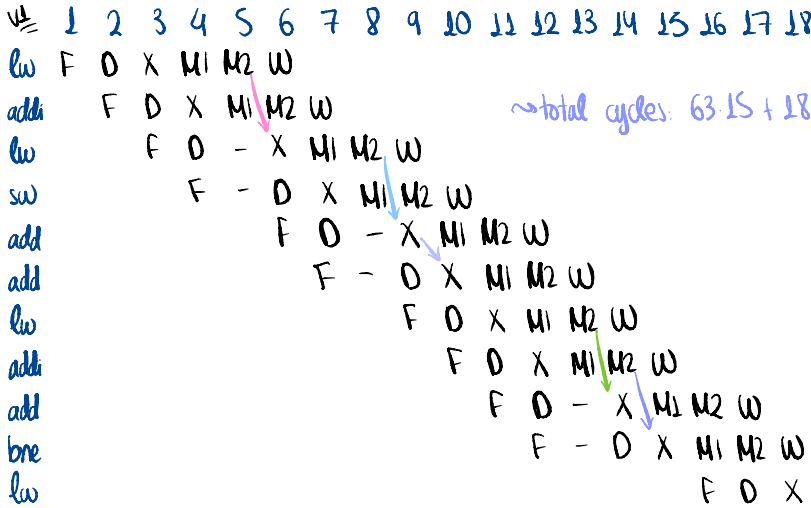
```
ADDI $t2, $t2, 4
LW $t1, 100($t2)
LW $t5, 0($t1)
SW $t1, 0($t2)
ADD $t3, $t3, $t5
ADD $t3, $t3, $t4
LW $t4, 0($t2)
ADD $t5, $t4, $t3
ADDI $t9, $t9, -4
BNE $t9, $t2, LOOP
```

LOOP:

```
LW $t1, 104($t2)
ADDI $t9, $t9, -4
LW $t5, 0($t1)
SW $t1, 4($t2)
ADD $t3, $t3, $t5
ADD $t3, $t3, $t4
LW $t4, 4($t2)
ADDI $t2, $t2, 4
ADD $t5, $t4, $t3
BNE $t9, $t2, LOOP
```

LOOP:

```
LW $t1, 104($t2)
ADDI $t2, $t2, 4
LW $t5, 0($t1)
SW $t1, 0($t2)
ADD $t3, $t3, $t5
ADD $t3, $t3, $t4
LW $t4, 0($t2)
ADDI $t9, $t9, -4
ADD $t5, $t4, $t3
BNE $t9, $t2, LOOP
```



Θέμα 3

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

- > Το πρόγραμμα εκτελείται σε έναν επεξεργαστή με ένα μόνο επίπεδο κρυφής μνήμης δεδομένων, η οποία αρχικά είναι άδεια. Η κρυφή μνήμη είναι 2-way associative, LRU, write-allocate με χωρητικότητα 512B και μέγεθος block 32 bytes. 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 ή επομένως αναφορά σε αυτές δεν συνεπάγεται πρόσβαση στην κρυφή μνήμη.
 Σε επίπεδο εντολών assembly οι αναγνώσεις γίνονται με τη σειρά που εμφανίζονται στον κώδικα.
 Οι πίνακες είναι ευθυγραμμισμένοι και αποθηκεύονται κατά γραμμές, ενώ συνεχής δόλωση μεταβλητών συνεπάγεται διαδοχική αποθήκευσή τους στη μνήμη.
 Α) Βρείτε το συνολικό αριθμό hits και misses για όλη την εκτέλεση του προγράμματος.

```
int i, j;
double A[16][8], B[16][8], C[8];

for (i = 0; i < 8; i++) {
    for (j = 0; j < 8; j++)
        A[i*4][j] = B[i*2][j] + A[i*4][j] + C[j];
}
```

Ζητούμενο 1

$$\begin{aligned} \text{block size} &= 32 \text{ bytes} \\ \text{στοιχείο size} &= 8 \text{ bytes} \end{aligned} \quad \left\{ \begin{aligned} \text{στοιχεία/block} &= 32/8 = 4 \\ \text{cache size} &= 512 \text{ bytes} = 2^9 \\ \text{block size} &= 32 \text{ bytes} = 2^5 \end{aligned} \right. \quad \left\{ \begin{aligned} \# \text{block}_1 &= 2^9/2^5 = 2^4 = 16 \\ \# \text{sets} &= 16/2 = 8 \end{aligned} \right.$$

set 0: A[0][0-3] A[4][0-3] B[...][0-3] C[0-3]

set 1: A[0][4-7] A[4][4-7] C[4-7]

set 2: A[1][0-3] A[5][0-3]

...

set 7: A[3][4-7] A[7][4-7]

i=0, j=0:

B[0][0] + A[0][0] + C[0] → A[0][0]

m m m h

LRU

C[0-3]	A[0][0-3]
--------	-----------

i=0, j=1-3

B[0][1] + A[0][1] + C[1] → A[0][1]

m h m h
 m h m h
 m h m h

LRU

C[0-3]	A[0][0-3]
--------	-----------

i=0, j=4-7

B[0][4] + A[0][4] + C[4] → A[0][4]

m m m h
 m h m h
 m h m h
 m h m h

LRU

C[0-3]	A[0][0-3]
C[4-7]	A[0][4-7]

i=1, j=0-3

B[1][0] + A[1][0] + C[1] → A[1][0]

m m h h
 h h h h
 h h h h
 h h h h

LRU

C[0-3]	A[0][0-3]
C[4-7]	A[0][4-7]
B[0][0-3]	A[1][0-3]

i=1, j=4-7

B[1][4] + A[1][4] + C[4] → A[1][4]

m m h h
 h h h h
 h h h h
 h h h h

LRU

C[0-3]	A[0][0-3]
C[4-7]	A[0][4-7]
B[1][0-3]	A[1][0-3]
B[1][4-7]	A[1][4-7]

$i=2, j=0-3$

$B(0)(0) + A(2)(0) + C(0) \rightarrow A(2)(0)$

m m h h
h h h h
h h h h
h h h h

C(0-3)	B(0)(0-3)
C(4-7)	B(0)(4-7)
B(2)(0-3)	A(1)(0-3)
B(2)(4-7)	A(1)(4-7)
A(2)(0-3)	

$i=2, j=4-7$

$B(0)(4) + A(2)(4) + C(4) \rightarrow A(2)(4)$

m m h h
h h h h
h h h h
h h h h

C(0-3)	B(0)(0-3)
C(4-7)	B(0)(4-7)
B(2)(0-3)	A(1)(0-3)
B(2)(4-7)	A(1)(4-7)
A(2)(0-3)	
A(2)(4-7)	

$i=3, j=0-3$

$B(1)(0) + A(3)(0) + C(0) \rightarrow A(3)(0)$

h m h h
h h h h
h h h h
h h h h

C(0-3)	B(0)(0-3)
C(4-7)	B(0)(4-7)
B(2)(0-3)	A(1)(0-3)
B(2)(4-7)	A(1)(4-7)
A(2)(0-3)	
A(2)(4-7)	
A(3)(0-3)	

$i=3, j=4-7$

$B(1)(4) + A(3)(4) + C(4) \rightarrow A(3)(4)$

h m h h
h h h h
h h h h
h h h h

C(0-3)	B(0)(0-3)
C(4-7)	B(0)(4-7)
B(2)(0-3)	A(1)(0-3)
B(2)(4-7)	A(1)(4-7)
A(2)(0-3)	
A(2)(4-7)	
A(3)(0-3)	
A(3)(4-7)	

$i=4, j=0-3$

$B(0)(0) + A(0)(0) + C(0) \rightarrow A(0)(0)$

h m m h
m h m h
m h m h
m h m h

A(0)(0-3)	C(0-3)
C(4-7)	B(0)(4-7)
B(2)(0-3)	A(1)(0-3)
B(2)(4-7)	A(1)(4-7)
A(2)(0-3)	
A(2)(4-7)	
A(3)(0-3)	
A(3)(4-7)	

$i=4, j=4-7$

$B(0)(4) + A(0)(4) + C(4) \rightarrow A(0)(4)$

h m m h
m h m h
m h m h
m h m h

A(0)(0-3)	C(0-3)
A(0)(4-7)	C(4-7)
B(2)(0-3)	A(1)(0-3)
B(2)(4-7)	A(1)(4-7)
A(2)(0-3)	
A(2)(4-7)	
A(3)(0-3)	
A(3)(4-7)	

Κοιτάστε

Ζητούμενο 2

B) Πώς θα επηρεαστούν τα hits και misses για όλη την εκτέλεση του παραπάνω κώδικα, χωρητικότητας αλλά 4-way associative; Πείτε όπως και πριν το συνολικό τους αριθμό.

Θα είχαμε μόνο compulsory misses = $8 + 4 + 2 = 12$ misses
access: $8 \cdot 4 = 256 \Rightarrow \text{hits} = 256 - 12 = 244$
 $G_i, G_j, G_{\text{access}}$

Θέμα 4

Ζητούμενο 1

```
int dotProduct {
    int res = 0;
    for (int i=0; i<n; i++){
        res+=v[i]*u[i];
    }
    return res;
}
```

FOR:

```
addi $sp, $sp, -4  $\rightarrow -8$ 
sw $s1, 4($sp)
sw $s2, 0($sp)
add $t1, $zero, $zero
add $t2, $zero, $zero
lw $s1, 0($a0)
lw $s2, 0($a1)
mul $s1, $s1, $s2
add $t1, $t1, $s1
addi $a0, $a0, 4
addi $a1, $a1, 4
addi $t2, $t2, 1
slt $t3, $t2, $a2
bne $t3, $zero FOR
add $v0, $zero, $t1
lw $s2, 0($sp)
lw $s1, 4($sp)
addi $sp, $sp, 4  $\rightarrow 8$ 
jr $ra
```

$\$a0 \rightarrow dv$
 $\$a1 \rightarrow du$
 $\$t1 \rightarrow res$ $\$t2 \rightarrow i$
 $\$s1 \rightarrow v[i]$
 $\$s2 \rightarrow u[i]$

Ζητούμενο 2

```
1. foo: addi $sp, $sp, -4  $\rightarrow -8$ 
2.      sw $s0, 4($sp)
3.      sw $s1, 0($sp)
4.      add $t1, $a1, $a0
5.      addi $s0, $t1, 10
6.      add $s2, $s0, $t1
7.      add $a0, $zero, $s2
8.      jal bar
9.      add $t2, $t1, $v0
10.     add $s1, $t2, $a1
11.     add $v0, $zero, $s1
12.     lw $s1, 0($sp)
13.     lw $s0, 4($sp)
14.     addi $sp, $sp, 4  $\rightarrow 8$ 
15.     jr $ra
```

$\rightarrow 4 \text{ addn}$