學號:B06902136
系級:資工四
姓名:賴冠毓

Part A: Handwritten
4.31
4.31.1

Cycle 1	li x12, 0 jal ENT
Cycle 2	bne x12, x13, TOP slli x5, x12, 3
Cycle 3	add x6, x10, x5 ld x7, 0(x6)
Cycle 4	ld x29, 8(x6) sub x30, x7, x29
Cycle 5	addi x31, x11, x5 sd x30, 0(x31)
Cycle 6	slli x5, x12, 3 nop
Cycle 7	add x6, x10, x5 nop
Cycle 8	ld x7, 0(x6) nop
Cycle 9	ld x29, 8(x6) nop
Cycle 10	sub x30, x7, x29 nop
Cycle 11	add x31, x11, x5 nop
Cycle 12	sd x30, 0(x31) addi x12, x12, 2
Cycle 13	bne x12, x13, TOP nop

4.31.2
原本跑 2 次迴圈要 21 個 cycles，2-issue 的話只需要 13 個 => speedup 為 1.62。
4.31.3

```
    bez x13, DONE
```

```
    li x12, 0
    jal ENT
TOP: ld x7, 0(x10)
    ld x29, 8(x10)
    addi x12, x12, 2
    sub x30, x7, x29
    sd x30, 0(x11)
    addi x10, x10, 16
    addi x11, x11, 16
ENT: bne x12, x13, TOP
DONE:
```

4.31.4

```
    beqz x13, DONE
    li x12, 0
TOP: ld x7, 0(x6)
    addi x12, x12, 2
    ld x29, 8(x6)
    addi x6, x6, 16
    sub x30, x7, x29
    sd x30, 0(x31)
    bne x12, x13, TOP
DONE:
```

4.31.5

Cycle 1	beqz x13, DONE nop
Cycle 2	li x12, 0 ld x29, 8(x6)
Cycle 3	addi x6, x6, 16 ld x7, 0(x6)
Cycle 4	add x31, x11, x5 nop
Cycle 5	sub x30, x7, x29 sd x30, 0(x31)
Cycle 6	bne x12, x13, TOP ld x7, 0(x6)
Cycle 7	addi x12, x12, 2 ld x29, 8(x6)
Cycle 8	addi x6, x6, 16

	nop
Cycle 9	sub x30, x7, x29 sd x30, 0(x31)
Cycle 10	bne x12, x13, TOP nop

4.31.6

原本每 2 個迴圈要跑 16 個 cycles，後來只需要 10 個 => speedup 為 1.6。

4.31.7

```
    beqz x13, DONE
    li x12, 0
TOP: slli x5, x12, 3
    add x6, x10, x5
    add x31, x11, x5
    ld x7, 0(x6)
    ld x29, 8(x6)
    ld x5, 16(x6)
    ld x15, 24(x6)
    addi x12, x12, 4
    sub x30, x7, x29
    sub x14, x5, x15
    sd x30, 0(x31)
    sd x14, 16(x31)
    bne x12, x13, TOP
DONE:
```

4.31.8

```
    beqz x13, DONE
    li x12, 0
    addi x6, x10, 0
TOP: ld x7, 0(x6)
    add x31, x11, x5
    ld x29, 8(x6)
    addi x12, x12, 4
    ld x16, 16(x6)
    slli x5, x12, 3
    ld x15, 24(x6)
    sub x30, x7, x29
    sd x30, 0(x31)
    sub x14, x16, x15
```

```
    sd x14, 16(x31)
    add x6, x10, x5
    bne x12, x13, TOP
DONE:
```

4.31.9

原本每 2 個迴圈要跑 15 個 cycles，後來只需要 10 個 => speedup 為 1.5。

4.31.10

```
    beqz x13, DONE
    li x12, 0
    addi x6, x10, 0
TOP: ld x7, 0(x6)
    add x31, x11, x5
    ld x29, 8(x6)
    addi x12, x12, 4
    ld x16, 16(x6)
    slli x5, x12, 3
    ld x15, 24(x6)
    sub x30, x7, x29
    sd x30, 0(x31)
    sub x14, x16, x15
    sd x14, 16(x31)
    add x6, x10, x5
    bne x12, x13, TOP
DONE:
```

原本每 2 個迴圈要跑 10 個 cycles，後來只需要 8 個 => speedup 為 1.25。

5.5

5.5.1

1 word = 4 bytes

$$\text{block size} = 2^{offset} = 2^5\ (bytes) = \frac{2^5}{4} = \frac{32}{4} = 8\ (words)$$

5.5.2

index bit 數 = 5

=> block 數 = $2^5 = 32$

5.5.3

data storage bits = block 數 × block size × 1 個 word 的 bit 數

$= 32 \times 8 \times (4 \times 8) = 8192$

total bits = data storage bits + tag bits + valid bits

$$= 8192 + 54 \times 32 + 1 \times 32 = 9952$$

$$\Rightarrow \text{ratio} = \frac{\text{total bits}}{\text{data storage bits}} = \frac{9952}{8192} = 1.215$$

5.5.4

Address		Tag (binary)	Index (binary)	Offset (binary)	hit/miss	bytes replaced (hex)
Hex	Dec					
00	0	0	00000	00000	miss	
04	4	0	00000	00100	hit	
10	16	0	00000	10000	hit	
84	132	0	00100	00100	miss	
E8	232	0	00111	01000	miss	
A0	160	0	00101	00000	miss	
400	1024	1	00000	00000	miss	00~1F
1E	30	0	00000	11110	miss	400~41F
8C	140	0	00100	01100	hit	
C1C	3100	11	00000	11100	miss	00~1F
B4	180	0	00101	10100	hit	
884	2180	10	00100	00100	miss	80~9F

5.5.5

$$\text{hit ratio} = \frac{hit}{hit + miss} = \frac{4}{12} = \frac{1}{3}$$

5.5.6

<index, tag, data>

<0, 3, MEM[0xC00]-MEM[0xC1F]>

<4, 2, MEM[0x880]-MEM[0x89F]>

<5, 0, MEM[0x0A0]-MEM[0x0BF]>

<7, 0, MEM[0x0E0]-MEM[0x0FF]>

5.10

5.10.1

$$clock\ rate_{P1} = \frac{1}{L1\ Hit\ Time_{P1}} = \frac{1}{0.66\ (ns)} = 1.52\ (\text{GHz})$$

$$clock\ rate_{P2} = \frac{1}{L1\ Hit\ Time_{P2}} = \frac{1}{0.90\ (ns)} = 1.11\ (\text{GHz})$$

5.10.2

$$Miss\ penalty = \left\lceil \frac{70}{0.66} \right\rceil = 107$$

$AMAT_{P1} = Hit\ time + Miss\ rate \times Miss\ penalty$

$= 1 + 0.08 \times 107 = 9.56$（cycles）

$Miss\ penalty = \left\lceil \frac{70}{0.90} \right\rceil = 78$

$AMAT_{P2} = Hit\ time + Miss\ rate \times Miss\ penalty$

$= 1 + 0.06 \times 78 = 5.68$（cycles）

5.10.3

$CPI_{P1} = base\ CPI + Miss\ rate \times Miss\ penalty$

$= 1 + 0.08 \times 107 + 0.36 \times 0.08 \times 107 = 12.64$

$CPI_{P2} = base\ CPI + Miss\ rate \times Miss\ penalty$

$= 1 + 0.06 \times 78 + 0.36 \times 0.06 \times 78 = 7.36$

=> P2 is faster.

5.10.4

$Miss\ penalty = \left\lceil \frac{5.62}{0.66} \right\rceil = 9$

$Extra\ miss\ penalty = \left\lceil \frac{70}{0.66} \right\rceil = 107$

$AMAT_{P1} = Hit\ time + Primary\ miss\ with\ L2\ hit + Primary\ miss\ with\ L2\ miss$

$= 1 + 0.08 \times 9 + 0.08 \times 0.95 \times 107 = 9.85$（cycles）

=> worse with L2 cache.

5.10.5

$CPI_{P1} = base\ CPI + Primary\ miss\ with\ L2\ hit + Primary\ miss\ with\ L2\ miss$

$= 1 + (0.08 \times 9 + 0.36 \times 0.08 \times 9)$

$+(0.08 \times 0.95 \times 107 + 0.36 \times 0.08 \times 0.95 \times 107) = 13.04$

5.10.6

$AMAT_{P1}$ with L2 cache $< AMAT_{P1}$ without L2 cache

$1 + 0.08 \times 9 + 0.08 \times$ L2 miss rate $\times 107 < 9.56$

=> L2 miss rate < 0.92

=> L2 miss rate $< 92\%$

5.10.7

P1、P2 的 cycle 大小不同

$AMAT_{P2} = 5.68$ (cycles) $= 5.68 \times 0.90 = 5.112$ (ns)

$AMAT_{P1}$ with L2 cache $< AMAT_{P2}$ without L2 cache

$1 + 0.08 \times 9 + 0.08 \times$ L2 miss rate $\times 107 < \frac{5.112}{0.66}$

=> L2 miss rate < 0.70

=> L2 miss rate $< 70\%$

5.16

5.16.1

1 page = 4KiB => 12 bits 為 page，virtual page 為前 4 bits.

Address		TLB hit/miss	page table hit/miss	page fault	TLB			
Decimal	hex				Valid	Tag	Physical Page Number	Time Since Last Access
4669	0x123d	miss	miss	Y	1	0xb	12	5
					1	0x7	4	2
					1	0x3	6	4
					1	0x1	13	0
2227	0x08b3	miss	hit	N	1	0x0	5	0
					1	0x7	4	3
					1	0x3	6	5
					1	0x1	13	1
13916	0x365c	hit	hit	N	1	0x0	5	1
					1	0x7	4	4
					1	0x3	6	0
					1	0x1	13	2
34587	0x871b	miss	miss	Y	1	0x0	5	2
					1	0x8	14	0
					1	0x3	6	1
					1	0x1	13	3
48870	0xbee6	miss	hit	N	1	0x0	5	3
					1	0x8	14	1
					1	0x3	6	2
					1	0xb	12	0
12608	0x3140	hit	hit	N	1	0x0	5	4
					1	0x8	14	2
					1	0x3	6	0
					1	0xb	12	1
49225	0xc040	miss	miss	Y	1	0xc	15	0
					1	0x8	14	3
					1	0x3	6	1
					1	0xb	12	2

5.16.2

1 page = 16KiB => 14 bits 為 page，virtual page 為前 2 bits，Tag bit 數變少
=> disadvantage: 重複使用 TLB 同一格，導致部分格子使用率下降
TLB 一格可以存一個 page，page size 變大則總共存的 physical memory 變多
=> advantage: TLB miss rate 下降

Address		TLB hit/miss	page table hit/miss	page fault	TLB			
Decimal	hex				Valid	Tag	Physical Page Number	Time Since Last Access
4669	0x123d	miss	hit	N	1	0xb	12	5
					1	0x7	4	2
					1	0x3	6	4
					1	0x0	5	0
2227	0x08b3	hit	hit	N	1	0xb	12	6
					1	0x7	4	3
					1	0x3	6	5
					1	0x0	5	0
13916	0x365c	hit	hit	N	1	0xb	12	7
					1	0x7	4	4
					1	0x3	6	6
					1	0x0	5	0
34587	0x871b	miss	miss	Y	1	0x2	13	0
					1	0x7	4	5
					1	0x3	6	7
					1	0x0	5	1
48870	0xbee6	hit	hit	N	1	0x2	13	0
					1	0x7	4	6
					1	0x3	6	8
					1	0x0	5	2
12608	0x3140	hit	hit	N	1	0x2	13	1
					1	0x7	4	7
					1	0x3	6	9
					1	0x0	5	0
49225	0xc040	hit	hit	N	1	0x2	13	2
					1	0x7	4	8
					1	0x3	6	0
					1	0x0	5	1

5.16.3

1 page = 4KiB => 12 bits 為 page，virtual page 為前 4 bits.

two − way set associative TLB => virtual page % 2，找對應的 set

再用 LRU 選要哪一格

=> 並且 Tag 為 $\left\lfloor \frac{\text{virtual page}}{2} \right\rfloor$

Address		TLB hit/miss	page table hit/miss	page fault	TLB				
Decimal	hex				Valid	Tag	Physical Page Number	Time Since Last Access	set
4669	0x123d	miss	miss	Y	1	0xb	12	5	0
					1	0x7	4	2	
					1	0x3	6	4	1
					1	0x0	13	0	
2227	0x08b3	miss	hit	N	1	0x0	5	0	0
					1	0x7	4	3	
					1	0x3	6	5	1
					1	0x0	13	1	
13916	0x365c	miss	hit	N	1	0x0	5	1	0
					1	0x7	4	4	
					1	0x1	6	0	1
					1	0x0	13	2	
34587	0x871b	miss	miss	Y	1	0x0	5	2	0
					1	0x4	14	0	
					1	0x1	6	1	1
					1	0x0	13	3	
48870	0xbee6	miss	hit	N	1	0x0	5	3	0
					1	0x4	14	1	
					1	0x1	6	2	1
					1	0x5	12	0	
12608	0x3140	hit	hit	N	1	0x0	5	4	0
					1	0x4	14	2	
					1	0x1	6	0	1
					1	0x5	12	1	
49225	0xc040	miss	miss	Y	1	0x6	15	0	0
					1	0x4	14	3	

					1	0x1	6	1	1
					1	0x5	12	2	

5.16.4

1 page = 4KiB => 12 bits 為 page，virtual page 為前 4 bits.

direct mapped TLB => virtual page % TLB 格數，找對應的 Index

=> 並且 Tag 為 $\left\lfloor \frac{\text{virtual page}}{\text{TLB 格數}} \right\rfloor$

Address		TLB hit/miss	page table hit/miss	page fault	TLB			
Decimal	hex				Valid	Tag	Physical Page Number	Index
4669	0x123d	miss	miss	Y	1	0xb	12	0
					1	0x0	13	1
					1	0x3	6	2
					0	0x4	9	3
2227	0x08b3	miss	hit	N	1	0x0	5	0
					1	0x0	13	1
					1	0x3	6	2
					0	0x4	9	3
13916	0x365c	miss	hit	N	1	0x0	5	0
					1	0x0	13	1
					1	0x3	6	2
					1	0x0	6	3
34587	0x871b	miss	miss	Y	1	0x2	14	0
					1	0x0	13	1
					1	0x3	6	2
					1	0x0	6	3
48870	0xbee6	miss	hit	N	1	0x2	14	0
					1	0x0	13	1
					1	0x3	6	2
					1	0x2	12	3
12608	0x3140	miss	hit	N	1	0x2	14	0
					1	0x0	13	1
					1	0x3	6	2
					1	0x0	6	3
49225	0xc040	miss	miss	Y	1	0x3	15	0
					1	0x0	13	1

					1	0x3	6	2
					1	0x0	6	3

5.16.5

如果沒有 TLB 的話，每次 memory access 都需要先經過 page table，才能再進到 physical memory 拿資料。有了 TLB 就比較快。

6.7

6.7.1

Core 執行順序	x	y	w = x + y + 1	z = x + y
1→2→3→4、 1→2→4→3、 2→1→3→4、 2→1→4→3	2	2	5	4
1→3→2→4、 2→3→1→4	2	2	3	4
1→3→4→2、 1→4→3→2、 2→3→4→1、 2→4→3→1	2	2	3	2
1→4→2→3、 2→4→1→3	2	2	5	2
3→1→2→4、 3→2→1→4	2	2	1	4
3→1→4→2、 3→2→4→1	2	2	1	2
3→4→1→2、 3→4→2→1、 4→3→1→2、 4→3→2→1	2	2	1	0
4→1→2→3、 4→2→1→3	2	2	5	0
4→1→3→2、 4→2→3→1	2	2	3	0

6.7.2

在每個 operation 做完後進行同步，讓每個 core 看到同一個變數的值都相同。

6.9

6.9.1

CPU SS: 每個 core 一次只能跑一個 thread。

Core1		Core2	
FU1	FU2	FU1	FU2
	A3	B1	B4
A1	A2	B1	B4
A1	A4	B2	
A1			B3

=> 花費 4 個 cycles，浪費 4 個 slots

6.9.2

CPU SS: 每個 core 一次只能跑一個 thread => 多的另一個 CPU 用不上。

CPU1				CPU2			
Core1		Core2		Core1		Core2	
	A3	B1	B4				
A1	A2	B1	B4				
A1	A4	B2					
A1			B3				

=> 花費 4 個 cycles，浪費 20 個 slots

6.9.3

CPU MT: 每個 cycle 只能處理來自同一個 thread 的指令。

FU1	FU2
A1	A2
A1	
A1	
A3	
A4	
B1	B4
B1	B4
B2	
B3	

=> 花費 9 個 cycles，浪費 6 個 slots

6.9.4

CPU SMT: 每個 cycle 能處理來自不同 thread 的指令。

FU1	FU2
A1	B1
A1	B1
A1	B2
A2	B3

A3	B4
A4	B4

=> 花費 6 個 cycles，浪費 0 個 slots

Part B: Programming

Part 1: Observing cache behavior

	dhrystone	median	multiply	qsort	rsort	towers	vvadd
Config 1	557936	8863	44964	269251	900737	7497	11830
Config 2	539075	8817	44947	257841	902477	7497	5053
Config 3	542214	8881	45032	257034	911861	7577	4808
Config 4	545513	8864	45111	254099	884849	7577	4653
Config 5	527386	8864	45112	254384	885937	7577	4653
Config 6	574790	8789	44900	269251	901048	7457	11830
Config 7	582962	8789	44892	269342	900876	7476	11808
Config 8	551369	9337	45091	274111	1025081	7485	12795
Config 9	551704	9315	45096	274363	1026321	7485	12872
Config 10	552352	9292	45101	274172	1026003	7499	13006
Config 11	546999	9390	45127	275235	1031835	7501	12648
Config 12	549202	9330	45112	263335	1051311	7606	5476
Config 13	547675	9361	45244	263814	1051300	7599	5541

Why are (1) the same or different?

different

config 1: Dcache = direct mapped

config 2: Dcache = 2-way

direct mapped，比較容易把舊資料洗掉，所以效率比較差。

Why are (2) the same or different?
different
config 3: random
config 4: LRU
LRU 有利於保存 Dcache 中比較常重複使用的資料。
Why are (3) the same or different?
Different
config 1: Dcache = direct mapped
config 3: Dcache = 4-way
random replace policy 造成用更多 way 而效率卻反而下降。
Why are (4) the same or different?
different
config 1: Icache = direct mapped
config 6: Icache = 2-way
config 7: Icache = 4-way
dhrystone 比較少重複使用同樣的指令，所以 association 越多，反而要多檢查格子。
Why are (5) the same or different?
different
config 12: bank = 1
config 13: bank = 4
因為 bank 數不同導致結果不同，不過 bank 並非影響 cycle 的主因，所以兩個雖不同但相差不大。
See the pmp.c in /root/emulator/benchmarks/pmp, what does this program want to do? And how does it make it?
pmp 是 physical memory protection。
跟 page table 有關，主要負責 virtual memory 轉到 physical memory 時不要超過要存取的範圍。
Change the number of cores available in crt.S file (line 125) in /root/emulator/benchmarks/common and recompile the mt-matmul program (for this question, matrix size is 32x32).
Report the cycle count of configuration17 on 1-core, configuration19 on 2-core, and configuration20 on 4-core.

configuration17 on 1-core	configuration19 on 2-core	configuration20 on 4-core
180192	92287	48239

Describe whether the cycle count decreases linearly, why or why not.
Yes，因為要執行的指令大多可以平行計算，所以隨著 core 數增加一倍，需要

的 cycle 數也會減少一半 => 線性遞減。

Part 2: Cache and matrix multiplication
Report on how you make your matrix multiplication and maybe some cache miss rate statistics using spike.
因為沒時間寫 inline assembly，所以選擇使用最原始的矩陣乘法。

Bonus: Architecture and Security
How to perform “exploiting conditional branch misprediction” attack?
用另一個外部的 process 傳進來一個 out of bound 的 x，然而系統沒有先檢查有沒有超出範圍，就直接將 x 轉換成 in bound 的 k，這樣 branch predictor 會判斷是 true，但實際上是 false，導致執行錯誤。
How to perform “poisoning indirect branches” attack?
搶在 function 執行前用大量錯誤的 data 去 mis-train branch predictor，之後遇到 function 時 branch predictor 會判斷執行前面 mis-train 的結果，而真正要被執行的 function 則會沒跑到，導致錯誤。
How to mitigate Spectre Attacks? (at least 3 methods)
1. 將外部來的 process 先隔離做特別處理。
2. 選擇性重新整理 TLB，可以減少修補攻擊造成的效能損失。
3. 對 CPU 架構跟操作模式設計進行修改。