NOTE: this is the SOLUTION to Exam 2.

The correct answers are indicated for each question, with explanations as needed.

Dr. Manikas

1 1 / 1 point

We are given the following MIPS code with the following latencies. What is the **baseline performance** (number of cycles) for this code?

Label	Opcode	Operands	Comment	Latency
Loop:	L.D	F8,0(Rx)	# F8 <= Mem[Rx+0]	2
I0:	MULT.D	F10,F0,F8	# F10 <= F0 * F8	4
I1:	L.D	F6,0(Ry)	# F6 <= Mem[Ry+0]	2
I2:	SUB.D	F4,F0,F8	# F4 <= F0 - F8	1
I3:	S.D	F4,0(Ry)	# Mem[Ry+0] <= F4	2
I4:	ADDI	Rx,Ry,#8	# Rx <= Ry + 8	0

✓ 17

Feedback

General Feedback

Examine the code and add latencies. Recall that latencies are **extra clock cycles needed**, so add 1 to each latency. Sum all values to get the baseline performance.

Label	Opcode	Operands	Comment	Latency	Cycles	Total Cycles

Loop:	L.D	F8,0(Rx)	# F8 <= Mem[Rx+0]	2	3	3
I0:	MULT.D	F10,F0,F8	# F10 <= F0 * F8	4	5	8
I1:	L.D	F6,0(Ry)	# F6 <= Mem[Ry+0]	2	3	11
I2:	SUB.D	F4,F0,F8	# F4 <= F0 - F8	1	2	13
I3:	S.D	F4,0(Ry)	# Mem[Ry+0] <= F4	2	3	16
I4:	ADDI	Rx,Ry,#8	# Rx <= Ry + 8	0	1	17

Thus, the total baseline performance of this code is 17 cycles.

2 1 / 1 point

We are given the following MIPS code. Registers that have **data dependencies** are (check all that apply):

Label	Opcode	Operands	Comment	Latency
Loop:	L.D	F2,0(Rx)	# F2 <= Mem[Rx+0]	3
I0:	MULT.D	F2,F0,F2	# F2 <= F0 * F2	4
I1:	L.D	F4,0(Ry)	# F4 <= Mem[Ry+0]	3
I2:	ADD.D	F4,F0,F4	# F4 <= F0 + F4	2
I3:	S.D	F4,0(Ry)	# Mem[Ry+0] <= F4	1
I4:	ADDI	Rx,Ry,#8	# Rx <= Ry + 8	0
I5:	SUB	R20,R4,Rx	# R20 <= R4 – Rx	0
I6:	BNZ	R20, Loop	# If R20 ≠ 0, goto Loop	1

- [x] F4
- [x] Rx
- [x] F2
- [ ] F0

- [x] R20
- [ ] Ry

Feedback

Based on answering correctly

F2 value updated in Loop: L.D, used as input in I0: MULT.D

F4 value updated in I1: L.D, used as input in I2: ADD.D

Rx value updated in I4: ADDI, used as input in I5: SUB

R20 value updated in I5: SUB, used as input in I6: BNZ

3 1 / 1 point

We are given the following partial MIPS code with the following latencies. How many **stalls** should be added after the first line of code?

Opcode	Operands	Latency
SUB.D	F4,F0,F8	1
S.D	F4,0(Ry)	2

- [ ] 2
- [ ] 4
- [x] 1
- [ ] 0
- [ ] 3

Feedback

General Feedback

S.D has a data dependency in register F4 with SUB.D.

SUB.D has a latency of 1, so we need to add 1 stall after instruction SUB.D.

4 1 / 1 point

We are given the following partial MIPS code with the following latencies. How many **stalls** should be added after the first line of code?

Opcode	Operands	Latency
L.D	F1,0(Rx)	3
MULT.D	F2,F0,F2	4

- 4
- 3
- ✓ 0
- 1
- 2

Feedback

General Feedback

There are no data dependencies between instructions L.D and MULT.D.

Therefore, no stalls are required: number of stalls should be 0.

5 1 / 1 point

We are given the following MIPS code with the following latencies:

		Latency
L.D	F2,0(R2)	2
MULT.D	F8,F6,F2	3
L.D	F4,4(R4)	2
ADD.D	F12,F0,F4	1
ADD.D	F10,F8,F2	1

The code has started executing using the following **Tomasulo's** architecture, up through clock cycle **3** as shown in the table below. Items updated during cycle 3 are shown in ***red*** font.

Instruction status table:

Instruction		j	k	Issue	Exec comp	Write result
L.D	F2	0	R2	1	3	
MULT.D	F8	F6	F2	2		
L.D	F4	4	R4	3		
ADD.D	F12	F0	F4			
ADD.D	F10	F8	F2			

Load buffers:

	Busy	Address
Load1	Yes	0+R2
Load2	Yes	4+R4
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MULT.D	F6			Load1
	Mult2	No					

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	F14
3	FU		Load1	Load2		Mult1			

During clock cycle 4, which **Load buffers** get updated (select all that are updated during this cycle)?

- [ ] Load2
- [x] Load1
- [ ] Load3
- [ ] No load buffers get updated

Feedback

General Feedback

- First L.D writes result to FU F2 and frees up Load Buffer Load1.
- MULT.D has both input register values (F6, F2) and now starts executing with latency 3.
- First ADD.D instruction is issued and occupies Reservation Station Add1.
- Register F0 is available now, but Add1 must wait for Load2 to get value for F4.

Load1 buffer is cleared, so this buffer gets updated.

Instruction status table:

Instruction		j	k	Issue	Exec comp	Write result
L.D	F2	0	R2	1	3	4
MULT.D	F8	F6	F2	2		
L.D	F4	4	R4	3		
ADD.D	F12	F0	F4	4		
ADD.D	F10	F8	F2			

Load buffers:

	Busy	Address
Load1	No	
Load2	Yes	4+R4
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	Yes	ADD.D	F0			Load2
	Add2	No					
	Add3	No					
3	Mult1	Yes	MULT.D	F6	F2		
	Mult2	No					

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	F14
4	FU		M[0+R2]	Load2		Mult1		Add1	

6 1 / 1 point

We are given the following MIPS code with the following latencies:

		Latency
L.D	F2,0(R2)	2
MULT.D	F8,F6,F2	3
L.D	F4,4(R4)	2
ADD.D	F12,F0,F4	1
ADD.D	F10,F8,F2	1

The code has started executing using the following **Tomasulo's** architecture, up through clock cycle **3** as shown in the table below. Items updated during cycle 3 are shown in ***red*** font.

Instruction status table:

Instruction		j	k	Issue	Exec comp	Write result
L.D	F2	0	R2	1	3	
MULT.D	F8	F6	F2	2		
L.D	F4	4	R4	3		
ADD.D	F12	F0	F4			
ADD.D	F10	F8	F2			

Load buffers:

	Busy	Address
Load1	Yes	0+R2
Load2	Yes	4+R4
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MULT.D	F6			Load1
	Mult2	No					

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	F14
3	FU		Load1	Load2		Mult1			

During clock cycle 4, which **Reservation Stations** get updated (select all that are updated during this cycle)?

- [ ] Add3
- [ ] Add2
- [ ] Mult2
- [ ] No reservation stations get updated
- [x] Add1
- [x] Mult1

Feedback

General Feedback

- First L.D writes result to FU F2 and frees up Load Buffer Load1.
- MULT.D has both input register values (F6, F2) and now starts executing with latency 3. Mult1 is updated.
- First ADD.D instruction is issued and occupies Reservation Station Add1.

- Register F0 is available now, but Add1 must wait for Load2 to get value for F4.

Both reservation stations Add1 and Mult1 are updated.

Instruction status table:

Instruction		j	k	Issue	Exec comp	Write result
L.D	F2	0	R2	1	3	4
MULT.D	F8	F6	F2	2		
L.D	F4	4	R4	3		
ADD.D	F12	F0	F4	4		
ADD.D	F10	F8	F2			

Load buffers:

	Busy	Address
Load1	No	
Load2	Yes	4+R4
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	Yes	ADD.D	F0			Load2
	Add2	No					
	Add3	No					
3	Mult1	Yes	MULT.D	F6	F2		
	Mult2	No					

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	F14
4	FU		M[0+R2]	Load2		Mult1		Add1	

7 1 / 1 point

We are given the following MIPS code with the following latencies:

		Latency
L.D	F2,0(R2)	2
MULT.D	F8,F6,F2	3
L.D	F4,4(R4)	2
ADD.D	F12,F0,F4	1
ADD.D	F10,F8,F2	1

The code has started executing using the following **Tomasulo's** architecture, up through clock cycle **3** as shown in the table below. Items updated during cycle 3 are shown in ***red*** font.

Instruction status table:

Instruction		j	k	Issue	Exec comp	Write result
L.D	F2	0	R2	1	3	
MULT.D	F8	F6	F2	2		
L.D	F4	4	R4	3		
ADD.D	F12	F0	F4			
ADD.D	F10	F8	F2			

Load buffers:

	Busy	Address
Load1	Yes	0+R2
Load2	Yes	4+R4
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MULT.D	F6			Load1
	Mult2	No					

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	F14
3	FU		Load1	Load2		Mult1			

During clock cycle 4, in the **Register result status** section, which registers get updated (select all that are updated during this cycle)?

- [ ] F8
- [ ] F4
- [x] F12
- [ ] F6
- [ ] F0
- [ ] F14
- [x] F2
- [ ] F10
- [ ] No registers get updated

Feedback

General Feedback

- First L.D writes result to FU F2 and frees up Load Buffer Load1.

- MULT.D has both input register values (F6, F2) and now starts executing with latency 3.
- First ADD.D instruction is issued and occupies Reservation Station Add1.
- Register F0 is available now, but Add1 must wait for Load2 to get value for F4.

Both registers F2 and F12 are updated.

Instruction status table:

Instruction		j	k	Issue	Exec comp	Write result
L.D	F2	0	R2	1	3	4
MULT.D	F8	F6	F2	2		
L.D	F4	4	R4	3		
ADD.D	F12	F0	F4	4		
ADD.D	F10	F8	F2			

Load buffers:

	Busy	Address
Load1	No	
Load2	Yes	4+R4
Load3	No	

Reservation Stations:

Time	Name	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	Yes	ADD.D	F0			Load2
	Add2	No					
	Add3	No					
3	Mult1	Yes	MULT.D	F6	F2		
	Mult2	No					

Register result status:

Clock		F0	F2	F4	F6	F8	F10	F12	F14
4	FU		M[0+R2]	Load2		Mult1		Add1	

8 1 / 1 point

A memory size of 1 MB = ___ bytes

- 2^{30}
- 2^{50}
- ✓ 2^{20}

- 2^{40}
- 2^{10}

Feedback

General Feedback

- 2^{10} = 1 K (kilo)
- 2^{20} = 1 M (mega)
- 2^{30} = 1 G (giga)
- 2^{40} = 1 T (tera)
- 2^{50} = 1 P (peta)

9 1 / 1 point

A memory size of 1 PB = ___ bytes

- 2^{20}
- 2^{40}
- 2^{10}
- ✓ 2^{50}
- 2^{30}

Feedback

General Feedback

- 2^{10} = 1 K (kilo)
- 2^{20} = 1 M (mega)
- 2^{30} = 1 G (giga)

- 2^{40} = 1 T (tera)
- 2^{50} = 1 P (peta)

10 1 / 1 point

A memory size of 1 GB = ___ bytes

- 2^{10}
- ✓ 2^{30}
- 2^{20}
- 2^{40}
- 2^{50}

Feedback

General Feedback

- 2^{10} = 1 K (kilo)
- 2^{20} = 1 M (mega)
- 2^{30} = 1 G (giga)
- 2^{40} = 1 T (tera)
- 2^{50} = 1 P (peta)

11 1 / 1 point

We are given a memory system with a 2-level hierarchy:

- Primary level access time = 5 ns
- Secondary level access time = 50 ns
- Hit rate = 80%

What is the **average access time** for this memory system?

✓ 14 ns

Feedback

General Feedback

For a two-level hierarchy, the average access time is given by:

$$t_a = ht_p + (1 - h)\, t_s$$

We are given

- Primary level access time t_p = 5 ns
- Secondary level access time t_s = 50 ns
- Hit rate = 80%, so h = 0.8

Then, average access time t_a is:

$$t_a = (0.8)\,(5\ ns) + (1 - 0.8)\,(50\ ns)$$
$$= (0.8)\,(5\ ns) + (0.2)\,(50\ ns)$$
$$= 4\ ns\ +\ 10\ ns\ =\ 14\ ns$$

12 1 / 1 point

We are given a memory system with a 2-level hierarchy:

- Primary level access time = 10 ns
- Secondary level access time = 100 ns
- Hit rate = 90%

What is the **average access time** for this memory system?

✓ 19 ns

Feedback

General Feedback

For a two-level hierarchy, the average access time is given by:

$$t_a = ht_p + (1 - h)\, t_s$$

We are given

- Primary level access time t_p = 10 ns
- Secondary level access time t_s = 100 ns
- Hit rate = 90%, so h = 0.9

Then, average access time t_a is:

$$t_a = (0.9)\,(10\ ns) + (1 - 0.9)\,(100\ ns)$$
$$= (0.9)\,(10\ ns) + (0.1)\,(100\ ns)$$
$$= 9\ ns\ +\ 10\ ns\ =\ 19\ ns$$

13 1 / 1 point

We are given a memory system with the following parameters:

- Main memory size is 16 GB
- Cache size is 2 MB
- Block size is 64 bytes

If the cache is **associative,** what are the number of bits for each field below?

TAG ✓ 28 BYTE ✓ 6

Feedback

General Feedback

An **associative** cache has two fields: tag (to identify main memory block) and byte (to identify byte in a block).

The main memory size = 16 GB = $2^4 2^{30}$ B = 2^{34} B

The block size is 64 B = 2^6 B, so the main memory is divided into $2^{34}/2^6 = 2^{28}$ blocks.

Thus, the memory address fields and corresponding number of bits are:

Tag	Byte
28	6

14 1 / 1 point

We are given a memory system with the following parameters:

- Main memory size is 64 GB
- Cache size is 128 MB
- Block size is 32 bytes

If the cache is **associative,** what are the number of bits for each field below?

TAG ✓ 31 BYTE ✓ 5

Feedback

General Feedback

An **associative** cache has two fields: tag (to identify main memory block) and byte (to identify byte in a block).

The main memory size = 64 GB = $2^6 2^{30}$ B = 2^{36} B

The block size is 32 B = 2^5 B, so the main memory is divided into $2^{36}/2^5 = 2^{31}$ blocks.

Thus, the memory address fields and corresponding number of bits are:

Tag	Byte
31	5

15 1 / 1 point

We are given a memory system with the following parameters:

- Main memory size is 4 GB
- Cache size is 64 MB
- Block size is 64 bytes

If the cache is **direct-mapped**, what are the fields in the memory address?

TAG ✓ 6 GROUP ✓ 20 BYTE ✓ 6

Feedback

General Feedback

For a **direct-mapped** cache, the main memory is viewed as table. The number of table rows = number of cache blocks, or **groups**

The main memory size = 4 GB = $2^2 2^{30}$ B = 2^{32} B

The block size is 64 B = 2^6 B, so the main memory is divided into $2^{32}/2^6 = 2^{26}$ blocks.

The cache size = 64 MB = $2^6 2^{20}$ B = 2^{26} B = $2^{26}/2^6 = 2^{20}$ blocks, or 2^{20} **groups (rows)**

The number of table columns = **tags** = # main memory blocks / # cache lines = $2^{26}/2^{20} = 2^6$

The resulting memory address fields are:

Tag	Group	Byte
6	20	6

16 1 / 1 point

We are given a memory system with the following parameters:

- Main memory size is 256 MB
- Cache size is 4 KB
- Block size is 16 bytes

If the cache is **direct-mapped**, what are the fields in the memory address?

TAG ✓ 16 GROUP ✓ 8 BYTE ✓ 4

Feedback

General Feedback

For a **direct-mapped** cache, the main memory is viewed as table. The number of table rows = number of cache blocks, or **groups**

The main memory size = 256 MB = $2^8 2^{20}$ B = 2^{28} B

The block size is 16 B = 2^4 B, so the main memory is divided into $2^{28}/2^4 = 2^{24}$ blocks.

The cache size = 4 KB = $2^2 2^{10}$ B = 2^{12} B = $2^{12}/2^4 = 2^8$ blocks, or 2^8 **groups (rows)**

The number of table columns = **tags** = # main memory blocks / # cache lines = $2^{24}/2^8 = 2^{16}$

The resulting memory address fields are:

Tag	Group	Byte
16	8	4

17 1 / 1 point

We are given a memory system with a direct-mapped cache. This cache has the following memory address fields:

Tag	Group	Byte
6	20	6

If we change the cache to **2-way set-associative,** what are the fields in the memory address?

TAG ✓ 7 SET ✓ 19 BYTE ✓ 6

Feedback

General Feedback

Recall that for a **direct-mapped** cache, the main memory is viewed as table. For this cache:

- Tag field = 6 bits, so number of table columns = 2^6
- Group field = 20 bits, so number of table rows = 2^{20}
- Byte field = 6 bits; this will be the same field size for the set-associative cache

A **2-way set associative** cache expands on the direct-mapped scheme by also viewing the cache as a table. Here, 2 ways = 2 cache table columns.

Therefore, the cache now has $2^{20}/2^1 = 2^{19}$ cache rows, or **sets**. Recall that the number of cache rows = number of main memory rows.

Since the number of rows is divided by 2, the number of main memory columns must be multiplied by the same amount. Therefore, number of main memory columns, or **tags** = $(2^6)(2^1)= 2^7$.

The resulting memory address fields are:

Tag	Set	Byte
7	19	6

18 1 / 1 point

We are given a memory system with a direct-mapped cache. This cache has the following memory address fields:

Tag	Group	Byte
9	22	5

If we change the cache to **8-way set-associative,** what are the fields in the memory address?

TAG ✓ 12 SET ✓ 19 BYTE ✓ 5

Feedback

General Feedback

Recall that for a **direct-mapped** cache, the main memory is viewed as table. For this cache:

- Tag field = 9 bits, so number of table columns = 2^9
- Group field = 22 bits, so number of table rows = 2^{22}
- Byte field = 5 bits; this will be the same field size for the set-associative cache

An **8-way set associative** cache expands on the direct-mapped scheme by also viewing the cache as a table. Here, 8 ways = 8 cache table columns.

Therefore, the cache now has $2^{22}/2^3 = 2^{19}$ cache rows, or **sets**. Recall that the number of cache rows = number of main memory rows.

Since the number of rows is divided by 8, the number of main memory columns must be multiplied by the same amount. Therefore, number of main memory columns, or **tags** = $(2^9)(2^3) = 2^{12}$.

The resulting memory address fields are:

Tag	Set	Byte
12	19	5

19 1 / 1 point

We are given a memory system with separate caches for instructions (I-cache) and data (D-cache). The system has the following specifications:

- Base CPI = 2
- Load and stores are 40% of instructions
- I-cache miss rate = 6%
- D-cache miss rate = 8%
- Miss penalty = 100 cycles

What is the **actual CPI** for this system if we consider the effects of cache misses?

✓ 11.2

Feedback

General Feedback

First, look at the effects of the I-cache

CPI added due to cache misses = (% items accessed)(miss rate)(miss penalty)

% items accessed = 1, since each instruction fetch accesses the I-cache

So $CPI_{I\text{-cache}} = (1)(0.06)(100) = 6$

Next, look at the effects of the D-cache

Here, % items accessed = 0.40, since D-cache is only accessed 40% of the time (loads and stores)

So $CPI_{D\text{-}cache} = (0.40)(0.08)(100) = 3.2$

Actual CPI = base CPI + $CPI_{I\text{-}cache}$ + $CPI_{D\text{-}cache}$ = 2 + 6 + 3.2 = **11.2**

20 1 / 1 point

We have **a virtual memory** system with *eight* virtual pages and *four* physical page frames. The page size is 1024 bytes. The page table is set up as follows:

Page #	Present bit	Disk address	Page frame field
0	1	01001100111	11
1	1	01001100011	00
2	1	10001100111	01
3	0	01010100111	11
4	0	10011001110	01
5	1	01111100111	10
6	0	01001100001	11
7	0	11011100111	00

We are also given the following **virtual address: 1024**
This maps to the following **main memory address (select from options below):**

- 5120
- 1024
- 7168
- 3072
- 2048
- 4096
- 6144
- ✓ 0000

Feedback

General Feedback

Virtual memory has 8 pages, where the size of each page is 1 KB = 1024 B

Therefore, the virtual addresses are the following:

Virtual addresses	page #
0-1023	0
1024-2047	1
2048-3071	2
3072-4095	3
4096-5119	4
5120-6143	5
6144-7167	6
7168-8191	7

Physical memory has 4 page frames:

Page frame	Page frame number	Physical addresses
Page frame 0	00	0-1023
Page frame 1	01	1024-2047
Page frame 2	10	2048-3071
Page frame 3	11	3072-4095

Virtual address 1024 refers to virtual memory page #1

Page table entry:

Page #	Present bit	Disk address	Page frame field
1	**1**	**01001100011**	**00**

Present bit = 1, so data is valid. The page frame field = 00. Therefore, the main memory address is **0000.**

21 1 / 1 point

We have **a virtual memory** system with *eight* virtual pages and *four* physical page frames. The page size is 1024 bytes. The page table is set up as follows:

Page #	Present bit	Disk address	Page frame field
0	1	01001100111	11
1	1	01001100011	00
2	1	10001100111	01
3	0	01010100111	11
4	0	10011001110	01
5	1	01111100111	10
6	0	01001100001	11
7	0	11011100111	00

We are also given the following **virtual address: 2048**
This maps to the following **main memory address (select from options below):**

- 5120
- 7168
- 3072
- ✓ 1024
- 2048
- 0000
- 4096

6144

Feedback

General Feedback

Virtual memory has 8 pages, where the size of each page is 1 KB = 1024 B

Therefore, the virtual addresses are the following:

Virtual addresses	page #
0-1023	0
1024-2047	1
2048-3071	2
3072-4095	3
4096-5119	4
5120-6143	5
6144-7167	6
7168-8191	7

Physical memory has 4 page frames:

Page frame	Page frame number	Physical addresses
Page frame 0	00	0-1023

Page frame 1 01	1024-2047
Page frame 2 10	2048-3071
Page frame 3 11	3072-4095

Virtual address 2048 refers to virtual memory page #2

Page table entry:

Page #	Present bit	Disk address	Page frame field
2	**1**	**10001100111**	**01**

Present bit = 1, so data is valid. The page frame field = 01. Therefore, the main memory address is **1024.**

22 1 / 1 point

We have **a virtual memory** system with *eight* virtual pages and *four* physical page frames. The page size is 1024 bytes. The page table is set up as follows:

Page #	Present bit	Disk address	Page frame field
0	0	01001100111	11
1	0	01001100011	00
2	0	10001100111	01
3	1	01010100111	11
4	1	10011001110	01
5	1	01111100111	10
6	0	01001100001	11
7	1	11011100111	00

We are also given the following **virtual address: 7168**
This maps to the following **main memory address (select from options below):**

- ✓ ◉ 0000
- ○ 5120

- 2048
- 1024
- 4096
- 3072
- 6144
- 7168

Feedback

General Feedback

Virtual memory has 8 pages, where the size of each page is 1 KB = 1024 B

Therefore, the virtual addresses are the following:

Virtual addresses	page #
0-1023	0
1024-2047	1
2048-3071	2
3072-4095	3
4096-5119	4
5120-6143	5
6144-7167	6
7168-8191	7

Physical memory has 4 page frames:

Page frame	Page frame number	Physical addresses
Page frame 0	00	0-1023
Page frame 1	01	1024-2047
Page frame 2	10	2048-3071
Page frame 3	11	3072-4095

Virtual address 7168 refers to virtual memory page #7

Page table entry:

Page #	Present bit	Disk address	Page frame field
7	1	11011100111	00

Present bit = 1, so data is valid. The page frame field = 00. Therefore, the main memory address is **0000.**

23 1 / 1 point

We have **a virtual memory** system with *eight* virtual pages and *four* physical page frames. The page size is 1024 bytes. The page table is set up as follows:

Page #	Present bit	Disk address	Page frame field
0	1	01001100111	11
1	1	01001100011	00
2	1	10001100111	01
3	0	01010100111	11
4	0	10011001110	01
5	1	01111100111	10
6	0	01001100001	11
7	0	11011100111	00

We are also given the following **virtual address: 0000**
This maps to the following **main memory address (select from options below):**

- 1024
- 0000
- 4096
- ✓ 3072
- 5120
- 6144
- 2048
- 7168

Feedback

General Feedback

Virtual memory has 8 pages, where the size of each page is 1 KB = 1024 B

Therefore, the virtual addresses are the following:

Virtual addresses	page #
0-1023	0
1024-2047	1
2048-3071	2
3072-4095	3
4096-5119	4
5120-6143	5

6144-7167	6
7168-8191	7

Physical memory has 4 page frames:

Page frame	Page frame number	Physical addresses
Page frame 0	00	0-1023
Page frame 1	01	1024-2047
Page frame 2	10	2048-3071
Page frame 3	11	3072-4095

Virtual address 0000 refers to virtual memory page #3

Page table entry:

Page #	Present bit	Disk address	Page frame field
0	**1**	**01001100111**	**11**

Present bit = 1, so data is valid. The page frame field = 11. Therefore, the main memory address is **3072.**

24 1 / 1 point

We have a 256 KB 4-way set-associative L2 cache that stores data as 32-byte blocks. To fetch a block from main memory, it takes 4 clock cycles for the first 4 bytes of the block (latency), then 1 clock cycle per 4 bytes for the remaining bytes of the block. What is the **miss penalty** for the cache (clock cycles)?

11 clock cycles

Feedback

General Feedback

Recall that miss penalty is the number of clock cycles to fetch one block from main memory into the cache.

We have 32 bytes/block, so divide into 4-byte groups, or 32/4 = 8 groups per block.

First group takes 4 clock cycles, and the remaining 7 groups each take 1 clock cycle

So we have 4 + 7(1) = **11 clock cycles** to fetch one block. This is the **miss penalty** for this cache.

25 1 / 1 point

We have a 1 MB 2-way set-associative L2 cache that stores data as 64-byte blocks. To fetch a block from main memory, it takes 8 clock cycles for the first 8 bytes of the block (latency), then 1 clock cycle per 8 bytes for the remaining bytes of the block. What is the **miss penalty** for the cache (clock cycles)?

15 clock cycles

Feedback

General Feedback

Recall that miss penalty is the number of clock cycles to fetch one block from main memory into the cache.

We have 64 bytes/block, so divide into 8-byte groups, or 64/8 = 8 groups per block.

First group takes 8 clock cycles, and the remaining 7 groups each take 1 clock cycle

So we have 8 + 7(1) = **15 clock cycles** to fetch one block. This is the **miss penalty** for this cache.