

HOMEWORK VI

DUE : December 6, 2011

READ :

- i) Related portions of Chapter 5
- ii) Related portions of Appendix B
- iv) Related portions of Appendix C

ASSIGNMENT : There are **eight** questions seven of which are from Chapter V of the textbook.

Solve all homework and exam problems as shown in class and past exam solutions.

1) Consider the following EMY mnemonic machine language function :

100	LW	R8, 1000(R4)	# The LW accesses location 1000
104	LW	R9, 1004(R4)	# Read from array B, starting at 1004
108	LW	R10, 2000(R4)	# Read from array C, starting at 2000
10C	ADDI	R4, R4, 4	
110	ADD	R11, R10, R9	
114	ADDI	R8, R8, (-1) ₁₀	
118	SLTI	R12, R8, 0	
11C	BEQ	R12, R0, (-7) ₁₀	
120	SW	R11, 2FFC(R4)	# Store to array A, starting at 3000
124	JR	R31	
128	NOP		
---		----	
1000	0		
1004	?		# Starting element of array B
---	----		
2000	?		# Starting element of array C
---	----		
3000	?		# Starting element of array A

All addresses are **physical** addresses

The EMY computer has a memory hierarchy now. The addresses the CPU generates are virtual addresses. For simplicity, above, we show the **physical addresses**, after they are translated from virtual addresses.

a) The EMY computer has a physical Level 1 instruction cache and a physical Level 1 data cache each one of which is a 2KB cache with direct mapping and with write-back and write allocate. The block length is 16 bytes. The physical memory contains 16MBytes. For **all** memory

addresses generated, clearly show which instruction/data is mapped to which block of which cache.

Then, assume that it is a **cold** start and indicate chronologically if there is a cache miss or hit, if a block is replaced and if a block is written back to the physical memory, for all memory accesses generated :

Address	Physical Memory cache block #	Cache memory block #	Hit/Miss	Block replaced/ Block written back
100	16	I Cache 16	Miss	None
....

b) Assume for this part that the EMY CPU is **pipelined** as discussed in class. Assume that the L1 cache memories take one (1) clock period when there is a hit. When there is a miss, the latency is four (4) clock periods and transferring a 4-byte item takes 1 clock period each.

Based on part (a), clearly show in which clock period the last cycle of the instruction that follows the JR instruction is performed as done in class, i.e. together with all the necessary forwarding, register-reading-writing and resolved data hazard types as done in class.

c) The EMY main memory is 8-way low-order interleaved. Clearly show which instruction and data are stored in which location of which memory bank. Show how the cache blocks in part (a) are mapped to the memory banks.

d) The EMY virtual memory has 2^{32} bytes. Assume that the above subroutine starts at virtual address 400100 and R4 has 10000000. The page length of the EMY computer is 4KBytes. For the two caches, the EMY has two TLBs, each with 64 entries and 4-way block set associative mapping with FIFO. Assume that when the CPU makes memory references there are always TLB hits, i.e. the TLBs always contain the physical page number of virtual pages. Show which set and which block of which TLB contains the entries and also show the important fields of the page table entries **after** the code completes.

2) Solve Problem 5.4.1. (a) of Chapter V.

3) Solve Problem 5.4.2. (a) of Chapter V.

4) Solve Problem 5.4.3. (a) of Chapter V.

Assume that for each block we need four extra bits : Valid, Dirty, Read and Write bits.

5) Solve Problem 5.7.1. (b) of Chapter V.

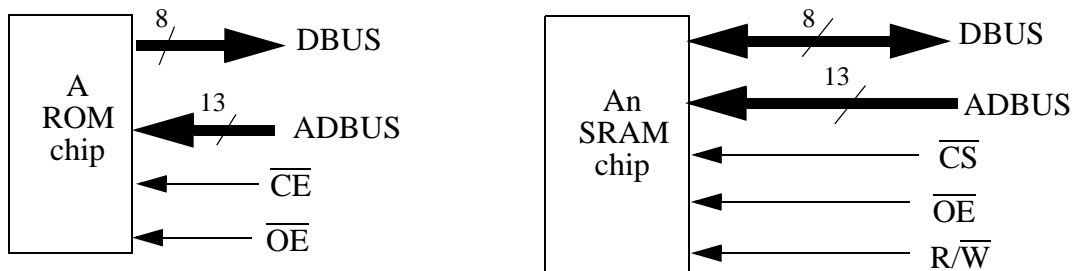
6) Solve Problem 5.7.2. (b) of Chapter V.

7) Solve Problem 5.7.3. (b) of Chapter V.

8) Solve Problem 5.10.4. (a) of Chapter V.

RELEVANT QUESTIONS AND ANSWERS

Q1) We are asked to design the memory subsystem of an imaginary computer. Its memory address space is **64KBytes**. We decide to use tri-state **ROM** and tri-state **SRAM** chips to implement the memory. We also decide that the size of the ROM area in the memory address space is **8 KBytes** and the remaining portion of the memory address space is for the SRAM area. We use the following ROM and SRAM chips :



Indicate the number of locations and the number of bits per location for the ROM and SRAM chips. Specify how many ROM and SRAM chips are used to implement the 64-KByte memory.

A1) ,, The ROM chip has 13 address lines, thus it has $2^{13} = 8192$ locations.
% The ROM has 8 data lines, thus it has 8 bits per location.
% Each one of the ROM chips is a 8Kx8-bit or 8Kx1-Byte or 8KByte chip.
% The ROM area size is 8KByte, thus we use only **1** ROM chip.

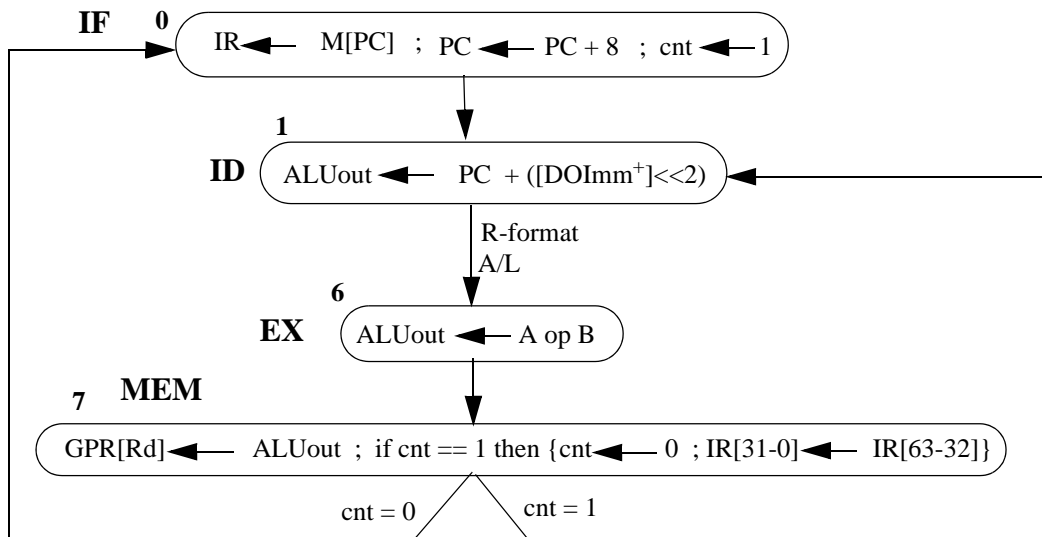
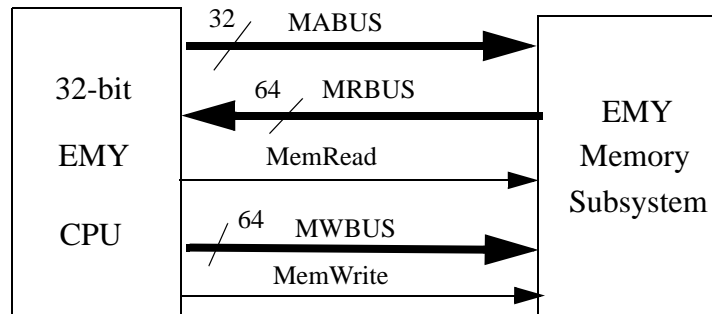
- „ The SRAM chip has 13 address lines, so it has $2^{13} = 8192$ locations.
- % The SRAM has 8 data lines and so it has 8 bits per locations.
- % Each one of the SRAM chips is a 8Kx1-Byte or 8KByte chip.
- % The size of the SRAM area is 64KByte - 8KByte = 56KByte.
- % Then, we need 56KByte/8KByte = 7 SRAM chips.

Q2) Assume that the EMY computer has a main memory that is **fast** and so there is **no** cache memory. Assume also that each EMY main memory location has 64 bits. The EMY CPU **architectural** registers are still 32 bits long. 32-bit instructions and data are stored in the same addresses as they are stored in the original 32-bit EMY memory.

Show how the signals between the EMY CPU and memory are changed. Modify the EMY high-level state diagram for the R-type A/L instructions only. What are their CPIs ?

A2) Each memory location has 64 bits and so has two instructions or two 32-bit data elements. This implies that we need to change those original states that have memory accesses : states 0, 3 and 5. We **also** need to change the last state of each instruction to decide if we have to fetch two instructions or continue with the next instruction that was already fetched. Overall, the number of memory accesses for the program will **decrease** by as much as a half.

The only modification between the CPU and the memory is the size of the data buses that must be 64 bits wide. The IR and MDR registers must be 64 bits long to be able to read 64 bits from the memory at once.



If the R-type A/L instruction is the first of the two instructions fetched then $CPI_{R\text{-type A/L}} = 4$ since we trace states 0, 1, 6 and 7.

If the R-type A/L instruction is the second of the two instructions fetched then $CPI_{R\text{-type A/L}} = 3$ since we trace states 1, 6 and 7.

This change is attractive since we reduce the number of memory accesses for the program and so the dependence on the memory. How would you modify the EMY CPU high-level state diagram for the remaining instructions ?

Q3) Assume that the EMY computer has a main memory that is **fast** and so there is **no** cache memory. Computer architects at our company have made a decision : the number of bits per EMY memory location is 16 bits while the EMY CPU architectural registers are still 32 bits long. 32-bit instructions and data are stored in the same addresses as they are stored in the original 32-bit EMY memory. Technically, this is mentioned as an internally 32-bit, externally 16-bit CPU. Commercial examples include the Motorola 68000 CPU.

Show the modified portion of the EMY high-level state diagram. What are the CPI s of the “LW” and “SW” instructions ?

A3) If each memory location has 16 bits, then each instruction is stored in two locations and also each data element is stored in two locations. This implies that we need to change those original states that have memory accesses : states 0, 3 and 5.

Now, each original memory access corresponds to two successive memory accesses. Therefore, each one of original states 0, 3 and 5 will be implemented by two states as shown on the next page.

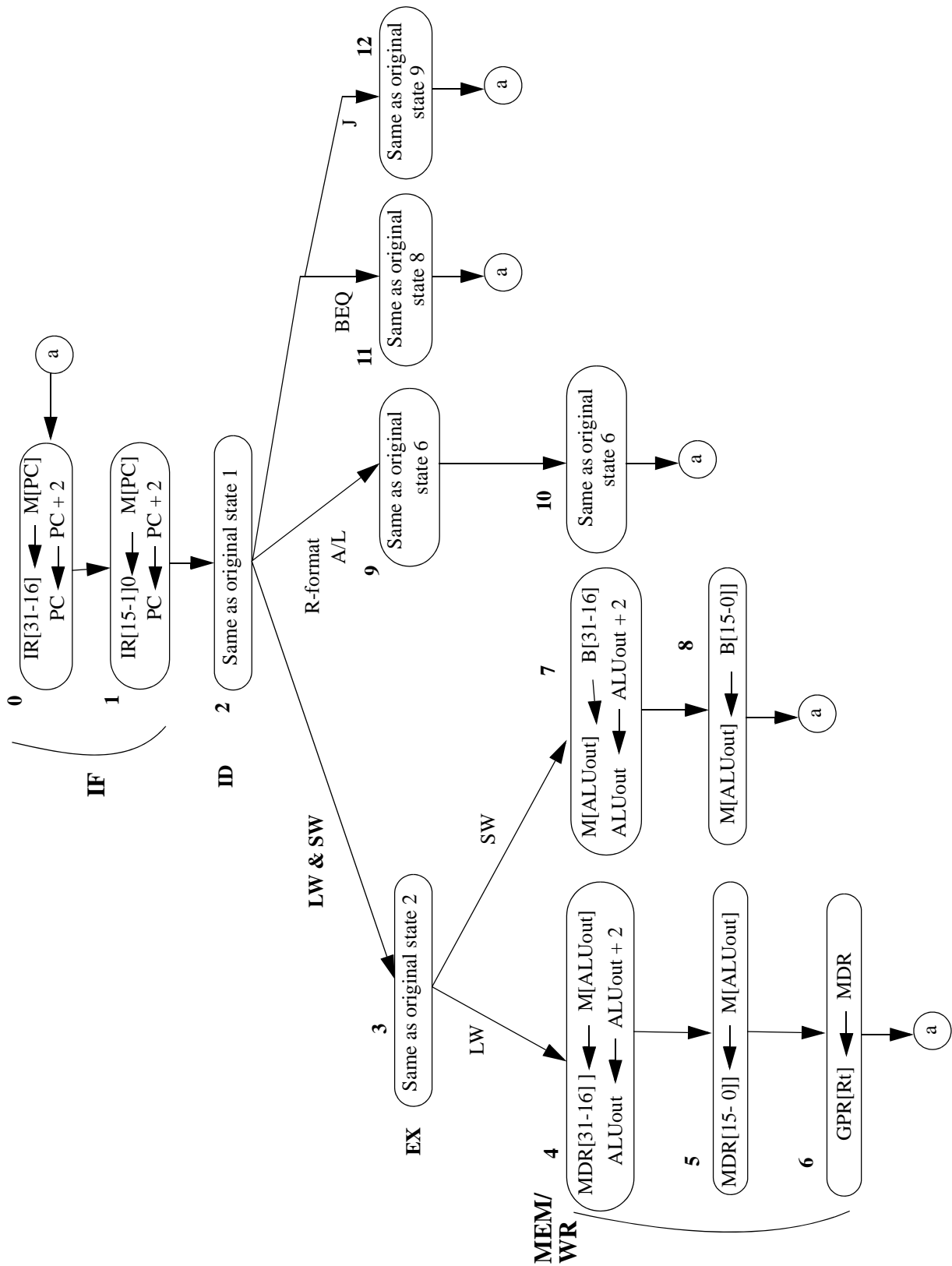
$CPI_{LW} = 7$ since we trace states 0, 1, 2, 3, 4, 5, 6.

$CPI_{SW} = 6$ since we trace states 0, 1, 2, 3, 7, 8.

The change results in the fact that for all instructions, the fetch cycle (the IF cycle) takes two clock periods as obvious from the state diagram on the next page. In addition, the MEM cycle of LW and SW instructions takes two clock periods.

Note that this change is attractive since we reduce the number of pins of the CPU chip which was important in the 1970s and 1980s. This idea can also be attractive today if a system has an 16-bit system (external) bus and the application needs the power of a 32-bit CPU.

The high-level state diagram is on the next page...



Q4) Consider the following EMY mnemonic machine language program :

```

400500 LW      R11, 0(R8)      # R8 initially has 10002200 and points at vector B
400504 LW      R12, 0(R9)      # R9 initially has 10003300 and points at vector C
400508 ADD     R13, R11, R12
40050C ADD     R14, R14, R13    # R14 is initially 0
400510 SW      R14, 0(R10)     # R10 initially has 10004400 and points at vector A
400514 ADDI    R8, R8, 4
400518 ADDI    R9, R9, 4
40051C ADDI    R10, R10, 4
400520 ADDI    R15, R15, (-1)10 # R15 determines the number of iterations
400524 BNE     R15, R0, (-10)10

```

Assume that the EMY computer has a **memory hierarchy** now.

a) Assume that $R15 = (100)_{10}$ initially, the miss ratio is 0.06 and the miss penalty is 20 clock periods. Calculate :

i) The CPI_{ave} for the program,

ii) The CPUtime if the clock frequency is 1 GHz, and

iii) $MIPS_{ave}$ for the program.

b) The addresses the CPU generates are virtual addresses. Thus, the addresses in Question 1 are all **virtual addresses**. Assume that

- The program starts at physical address 500,
- Data elements that start at virtual address 10002200 start at physical address 2200,
- Data elements that start at virtual address 10003300 start at physical address 3300, and
- Data elements that start at virtual address 10004400 start at physical address 4400.

The EMY computer has a physical Level 1 instruction cache and a physical Level 1 data cache each one of which is a 4KB cache with direct mapping and with write-back and write allocate. The block length is 16 bytes. The physical memory contains 16MBytes.

Assume that **R15 is 1 initially** :

i) **Clearly** show which instruction/data are mapped to which block of which cache.

ii) Then, assume that it is a **cold** start and, indicate chronologically if there is a cache miss or hit, if a block is replaced and if a block is written back to the physical memory, for all memory accesses generated :

Address	Physical memory block #	Which cache, Which block	Hit/Miss	Block replaced/ Block written back
500	80	I Cache 80	Miss	None
...

Is the **miss ratio really 0.06** over 100 iterations ? Is it possible to determine the miss ratio from part (ii) ? Explain.

A4) a)

i) The CPI_{ave} for the program : First we have to obtain the total number of memory accesses :

LW and SW instructions require 2 memory accesses each. The other instructions require 1 memory access. For a single iteration, we have then 13 memory accesses. For 100 iterations, there are 1300 memory accesses.

The average memory stall cycles per instruction : The total number of memory accesses is 1300. Then, the number of memory stall cycles for the program is $1300 * 0.06 * 20 = 1560$. Then, the average memory stall cycles per instruction is $1560/1000 = 1.56$.

The CPI_{ave} is increased by 1.56 compared with the ideal-memory case : $4.1 + 1.56 = \mathbf{5.66}$

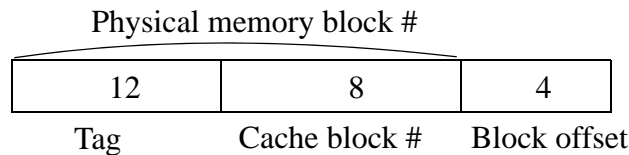
ii) The CPUtime : The total number of clock periods for the program is $5.66 * 1000 = 5660$ clock periods. Then :

$$\begin{aligned} \text{CPUtime} &= \text{clock periods for the program} \times \text{clock period duration} \\ &= 5660 \times 1 \times 10^{-9} \text{ seconds} = \mathbf{5.66 \text{ microseconds}} \end{aligned}$$

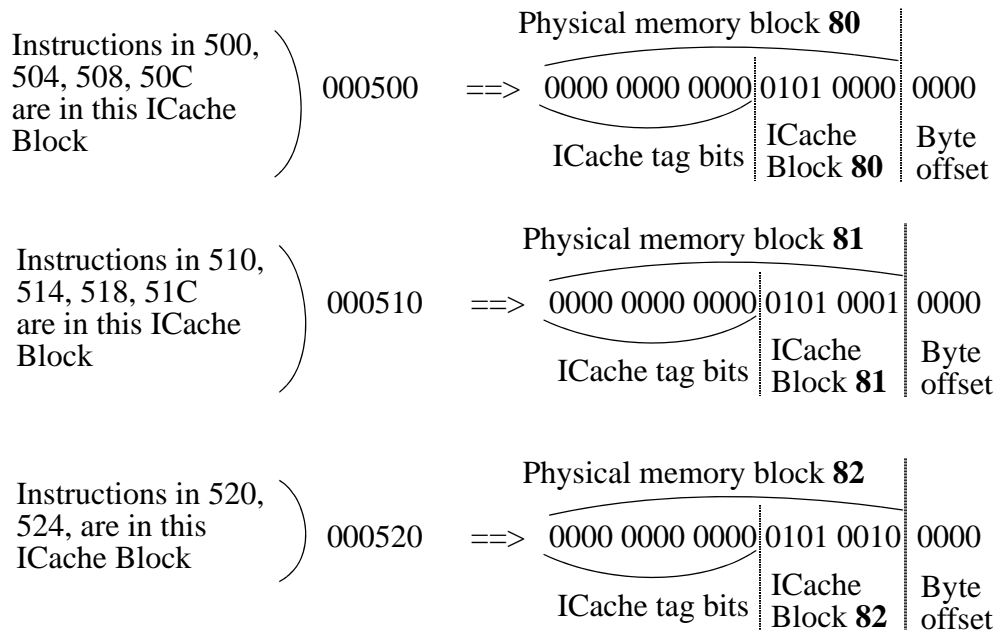
iii) $MIPS_{ave}$ for the program :

$$MIPS_{ave} = \frac{NI}{\text{CPUtime} \times 10^6} = \frac{1000}{5.66 \times 10^{-6} \times 10^6} = \mathbf{176.68}$$

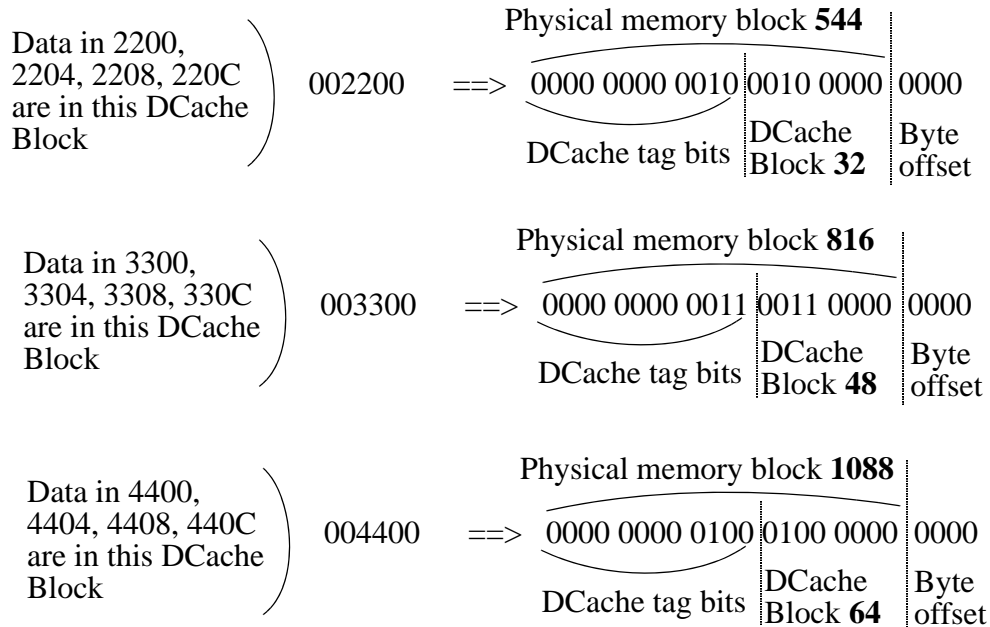
b) The number of cache blocks = (cache size in bytes)/(block size in bytes) = $2^{12}/2^4 = 2^8 = 256$ blocks in each cache. The physical address from the point of cache blocks is as follows :



The instruction cache, the ICache :



The data cache, the DCache :



Address	Physical memory block #	Which cache, Which block	Hit/Miss	Block replaced/ Block written back
500	80	I Cache 80	Miss	None
2200	544	D Cache 32	Miss	None
504	80	I Cache 80	Hit	None
3300	816	D Cache 48	Miss	None
508	80	I Cache 80	Hit	None
50C	80	I Cache 80	Hit	None
510	81	I Cache 81	Miss	None
4400	1088	D Cache 64	Miss	None
514	81	I Cache 81	Hit	None
518	81	I Cache 81	Hit	None
51C	81	I Cache 81	Hit	None
520	82	I Cache 82	Miss	None
524	82	I Cache 82	Hit	None

We see that the instructions occupy three blocks and we experience only three misses for the three blocks for the execution of the entire loop. Thus, for 1000 instruction executions, we have just three misses.

We also see that we experience one miss for every four data elements of each vector. For 100 iterations, 100 data ele-

ments are accessed per vector. Therefore, 25 misses per vector will be experienced. Thus, the total number of misses for the three vectors on 300 data accesses will be 75. In addition, the data accesses do not interfere with each other. Then, we can determine the miss ratio.

Then, we have a total number of 78 misses for 1300 memory accesses. The miss ratio is $178/1300 = 0.06$. Yes, the given miss ratio in part (a) is correct.

Q5) The EMY computer has a memory hierarchy that is as follows : there are two 2KB L1 physical cache memories with 16-byte blocks, direct mapping and write-back with write allocate, an 8-way low-order interleaved 2^{24} -byte physical memory, a 2^{32} -byte virtual memory with 2^{12} -byte pages and two TLBs each with 4-way block-set associative mapping and 64 entries.

Consider the following piece of EMY mnemonic language program :

```

400200 LWC1    F0, 0(R8)           # R8 initially has 10001400
400204 LWC1    F2, 0(R9)           # R9 initially has 10002400
400208 LWC1    F4, 0(R10)          # R10 initially has 10003400
40020C DIV.S   F6, F0, F2
400210 MUL.S   F8, F6, F4
400214 ADDI    R8, R8, 4
400218 ADDI    R9, R9, 4
40021C ADDI    R10, R10, 4
400220 ADDI    R11, R11, (-1)10    # R11 initially has 2
400224 ADD.S   F10, F8, F10        # F10 has already been initialized
400228 SWC1    0(R10), F10
40022C BNE    R11, R0, (-12)10

```

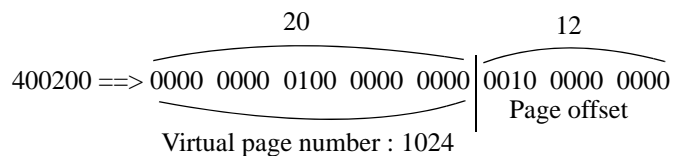
You are given that the operating systems maps

- „ virtual page 1024 (400 in Hex) to physical page 6,
- „ virtual page 65537 (10001 in Hex) to physical page 7,
- „ virtual page 65538 (10002 in Hex) to physical page 8 and
- „ virtual page 65539 (10003 in Hex), to physical page 9.

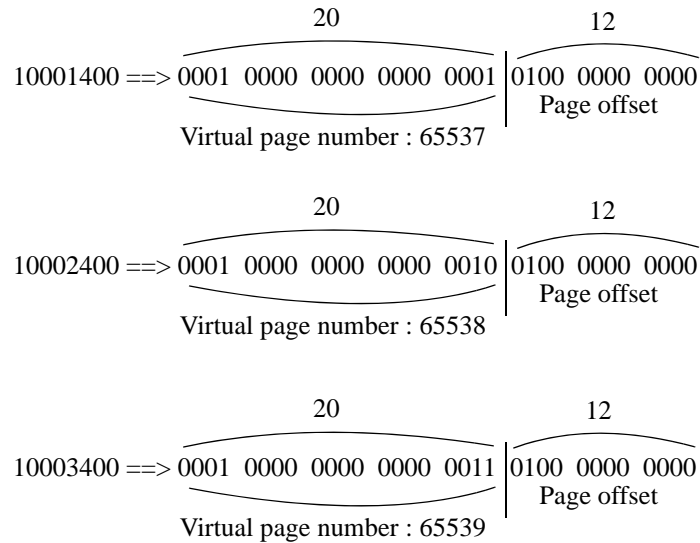
Assume that for this problem, the caches are initially empty. However, the TLBs do not generate misses and the physical memory does not result in a page fault.

Show which instruction and data are in which cache, which cache block and which physical memory block. Also, show which set and which block of which TLB contains the entries and also show the important fields of the page table entries after the code completes.

A5) By inspecting the program, it is observed that the instructions are in virtual page 1024 :



The data elements accessed are in virtual pages 65537, 65538 and 65539 :



Next, instruction addresses are inspected :

„ 400200-400234, all in virtual page 1024. Then these instructions are in 6200-6234, all in physical page 6.

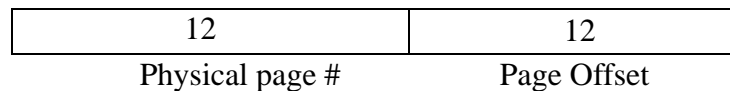
We now inspect data addresses :

„ 10001400-10001404, all in virtual page 65537. Then, these data elements are in 7400-7404, all in physical page 7.

„ 10002400-10002404, all in virtual page 65538. Then these data elements are in 8400-8404, all in physical page 8.

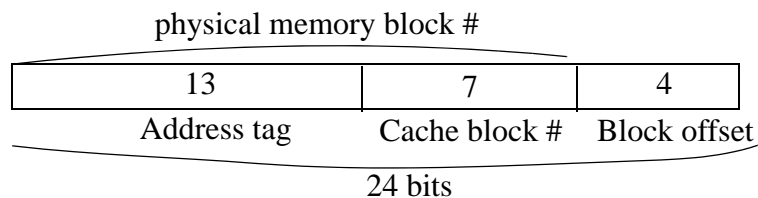
„ 10003400-10003404, all in virtual page 65539. Then these data elements are in 9400-9404, all in physical page 9.

We observe the fields of the 24-bit physical address :

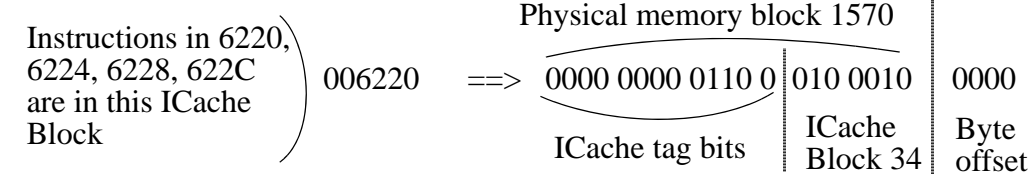
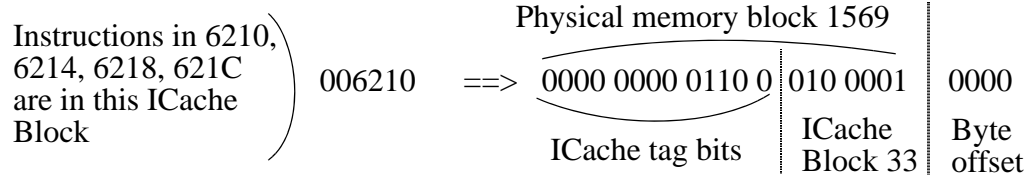
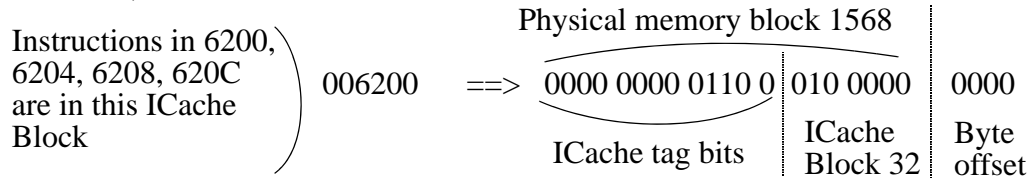


Next, the number of cache blocks = (cache size in bytes)/(block size in bytes) = $2^{11}/2^4 = 2^7 = 128$

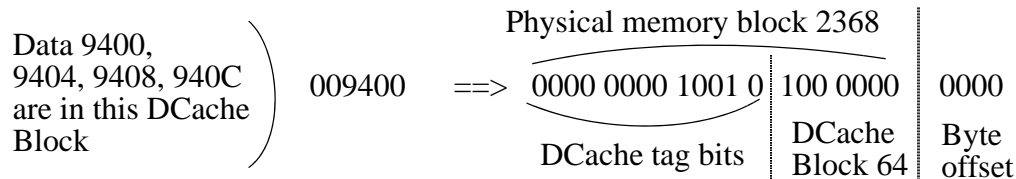
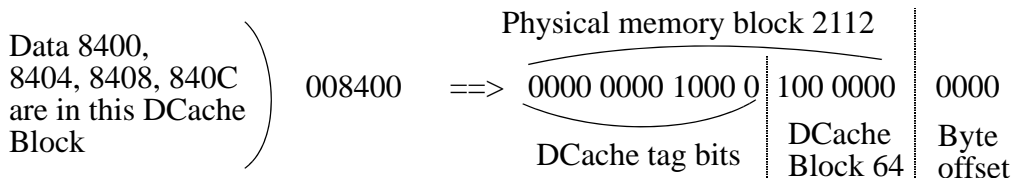
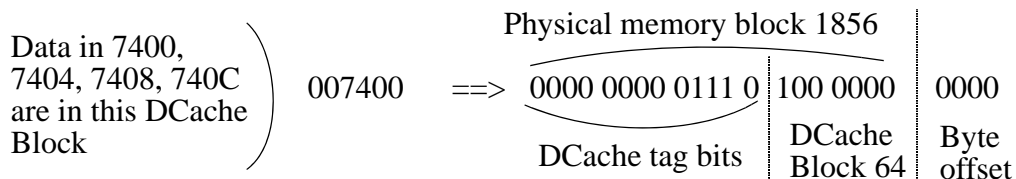
Now, we observe the physical address from the point of the cache blocks :



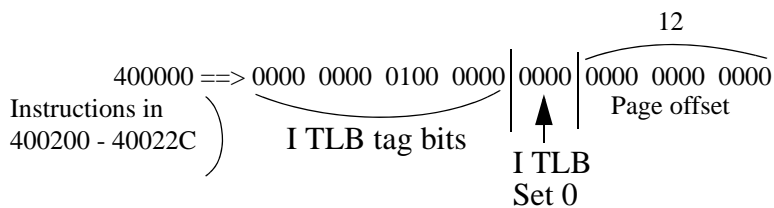
Now, the instruction cache, the ICache :



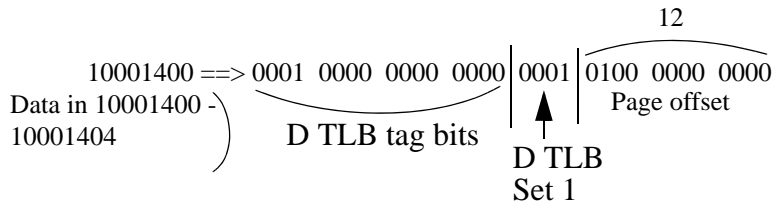
Now, the data cache, the DCache :



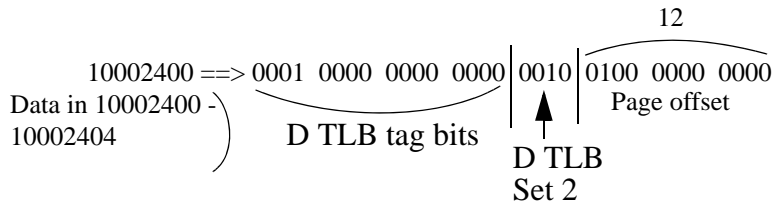
All instructions are on **virtual** page 1024 and use Instruction TLB set 0 :



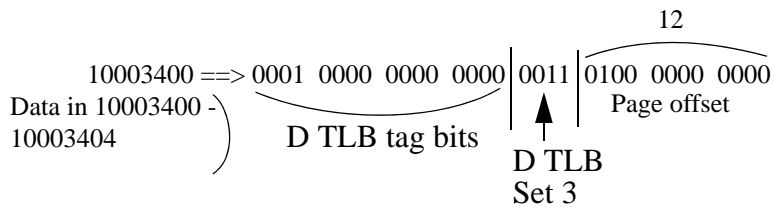
Data in 10001400-10001404 are on **virtual** page 65537 and use Data TLB set 1 :



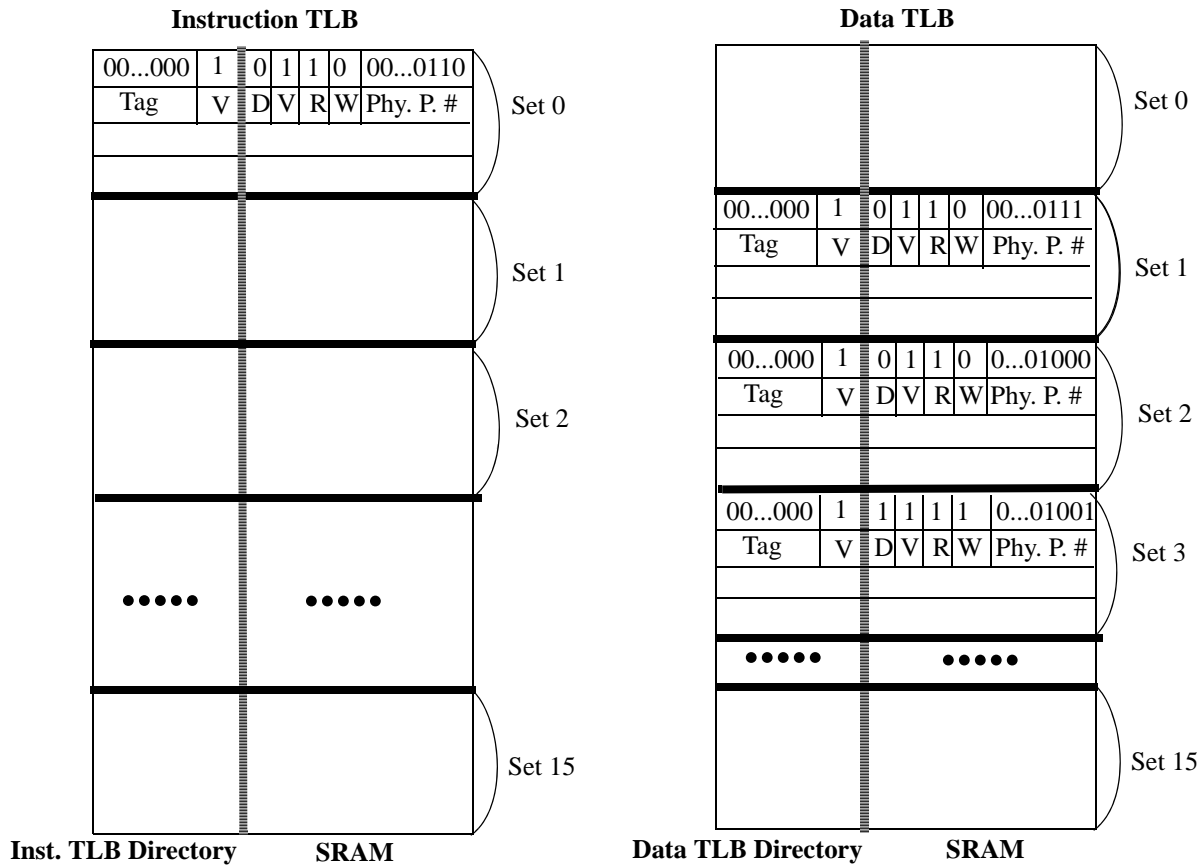
Data in 10002400-10002404 are on **virtual** page 65538 and use Data TLB set 2 :



Data in 10003400-10003404 are on **virtual** page 65539 and use Data TLB set 3 :



The TLB entries



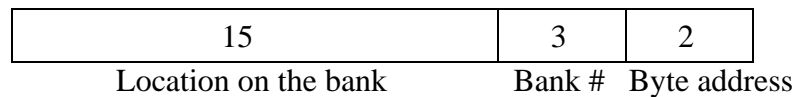
Q6) The EMY CPU has a memory hierarchy. There are physical instruction and data cache memories. The caches use direct mapping with write-back and write allocate. The latency of the physical memory is eight clock periods and there is an additional one clock period transfer time per word of the block. The physical memory is low-order interleaved. The size of the physical memory is 2^{20} bytes. The TLB size is 64 entries with fully-associative mapping and FIFO. The page size is 2^{12} bytes.

In order to speed up the L1 cache hit time, we decide to start the TLB access and the L1 cache access simultaneously even though the caches are physical. Determine L1 cache memory sizes and block sizes. Design the physical memory : show how physical addresses are mapped to the banks. Also, for **all** memory addresses generated, show where the following instructions and data are mapped in the physical memory. The addresses the CPU generates are virtual addresses. For simplicity, below, we show the **physical addresses**, after virtual addresses are translated.

500	LWC1	F1, 0(R4)	# LWC1 accesses location 2004	All addresses are physical addresses
504	MUL.S	F3, F2, F1	# F2 is passed a value	
508	DIV.S	F2, F3, F4	# F4 is passed a value	
50C	ADDI	R4, R4, 4		
510	ADDI	R5, R5, (-1) ₁₀	# R5 is passed a value of 2	
514	BNE	R5, R0, (-6) ₁₀		
518	SWC1	F2, 0(R6)	# SWC1 stores to location 2000	
51C	JR	R31		

A6) Since the physical memory latency is 8 clock periods, the number of physical memory banks must be greater than or equal to the memory latency in clock periods. Then, the number of banks is at least 8. We decide to have 8 banks.

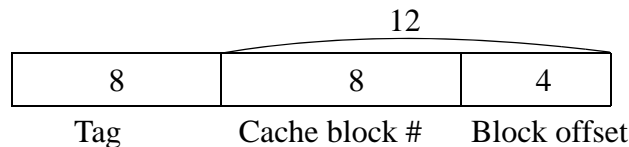
The physical memory address fields with 8-way low-order interleaving are as follows :



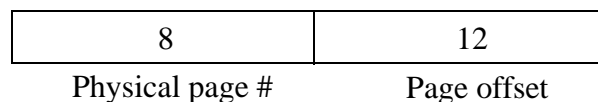
We know that each cache block must be distributed across the memory banks, so the block size must be either 4 or 8 words.

We decide to have 4 words per block : each L1 cache has $2^{12}/2^4 = 2^8 = 256$ blocks.

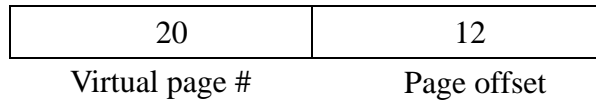
The physical address from the point of cache blocks is as follows :



The physical address from the point of physical pages :



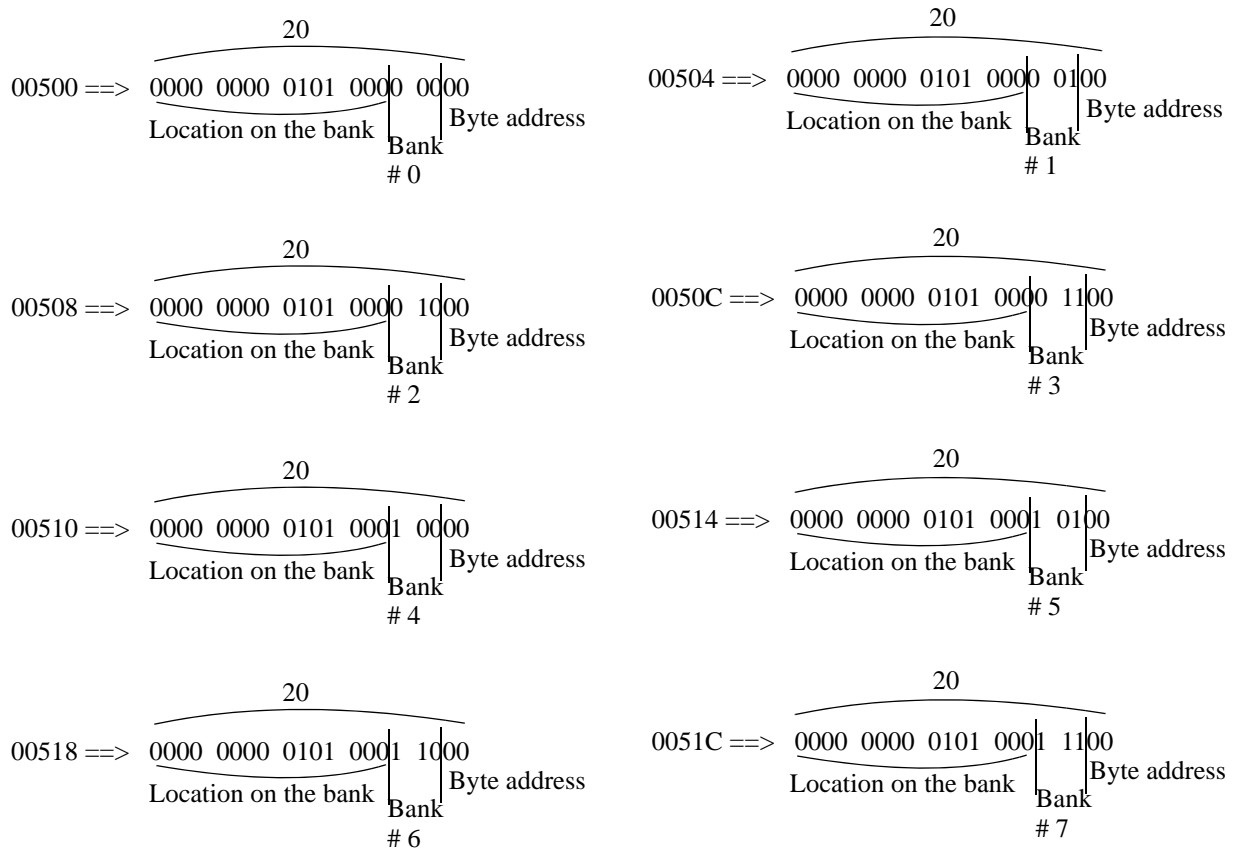
The virtual address from the point of virtual pages :



For simultaneous TLB and cache accesses to happen, the cache memory size must be the same as the page size. In the case of direct mapping :

$$\text{page size} = 2^{12} = 4096 \text{ bytes} = \text{cache memory size}$$

Thus, the lower 12 bits, the page offset, are the same for virtual and physical addresses. So, while the cache is working on the 8-bit cache block number to locate and retrieve the block in the SRAM portion and the tag bits in the directory portion, the TLB converts the 20-bit virtual page number (the leftmost 20 bits of the virtual address) to an 8-bit physical number. All that is left is the matching of the eight tag bits of the block and the 8-bit physical page number. The mapping of the instructions and data of the above program to the physical memory banks is as follows :



The memory banks are as follows :

Bank #	0	1	2	3	4	5	6	7
Physical addresses Their content	0 ?	4 ?	8?	C ?	10 ?	14 ?	18 ?	1C ?
Physical addresses Their content
Physical addresses Their content	500 LWC1 F1, 0(R4)	504 MUL.S F3, F2, F1	508 DIV.S F2, F3, F4	50C ADDI R4, R4, 4	510 ADDI R5, R5, (-1) ₁₀	514 BNE R5, R0, (-6) ₁₀	518 SWC1 F2, 0(R6)	51C JR R31
Physical addresses Their content
Physical addresses Their content	2000 (SWC1 stores here)	2004 (LWC1 reads here 1 st time)	2008 (LWC1 reads here 2 nd time)	200C	2010	2014	2018	201C
Physical addresses Their content
Physical addresses Their content	FFFE0	FFFE4	FFFE8	FFFE C	FFFF0	FFFF4	FFFF8	FFFFC

Q7) The EMY computer has a 32KByte **second** level unified cache. The physical memory is 2^{24} bytes. The block size is 128 bytes. The L2 cache uses 4-way block-set associative mapping with write-back, write allocate and **LRU** replacement. The addresses the CPU generates are virtual addresses. For simplicity, below, we show the **physical addresses**, after virtual addresses are translated.

2000	LW	R8, 0(R4)	# The LW accesses 8000, containing array A	All addresses are physical addresses
2004	SLT	R9, R8, R5	# R5 is initialized to 0	
2008	BEQ	R9, R0, 4		
200C	LW	R10, 2000(R4)	# The LW accesses A000 the first time	
2010	LW	R11, 4000(R4)	# The LW accesses C000 the first time	
2014	SW	R11, 2000(R4)		
2018	SW	R10, 4000(R4)		
201C	ADDI	R4, R4, 4	# Update the pointer register	
2020	ADDI	R6, R6, (-1) ₁₀	# R6 has 2 initially	
2024	BNE	R6, R0, (-10) ₁₀		
2028	JR	R31		

For **all** memory addresses generated, map the instructions and data elements of the above function to the secondary cache and the physical memory for the all the iterations of the loop. Clearly show why !

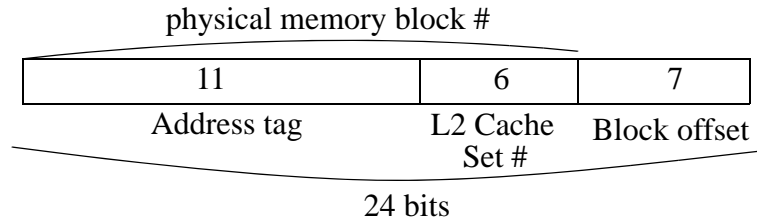
Then, assume that array A elements are all **negative** and it is a **cold** start, indicate in the chronological order, if there is a cache miss or hit, if a block is replaced and a block is written back to the physical memory for all memory accesses generated :

Address	Physical Memory cache block #	Cache memory set #/block #	Hit/Miss	Block replaced/ Block written back
2000	64	I Cache 0/0	Miss	None
....

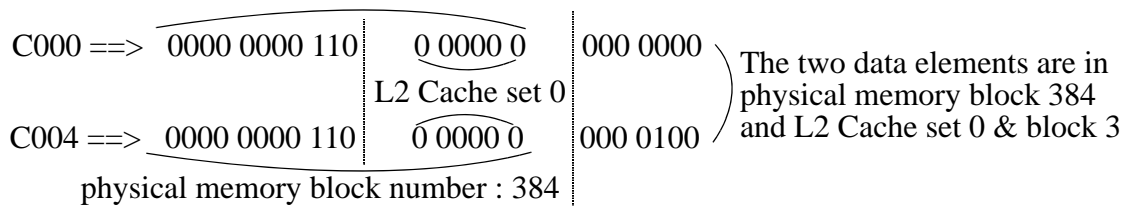
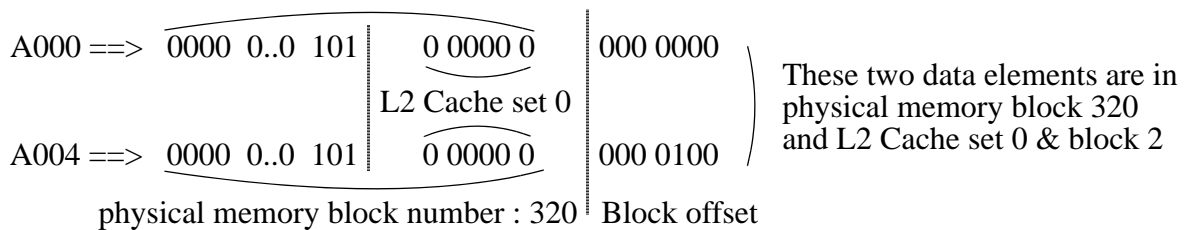
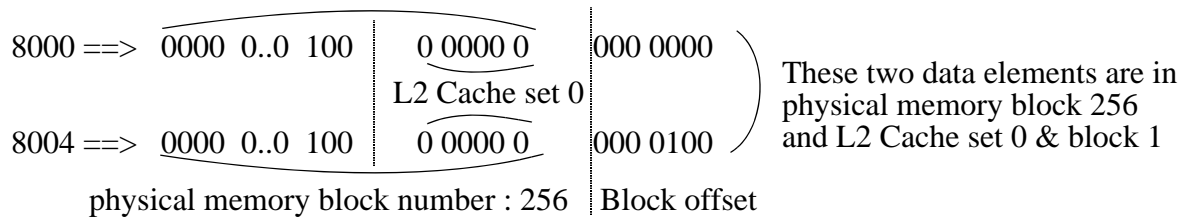
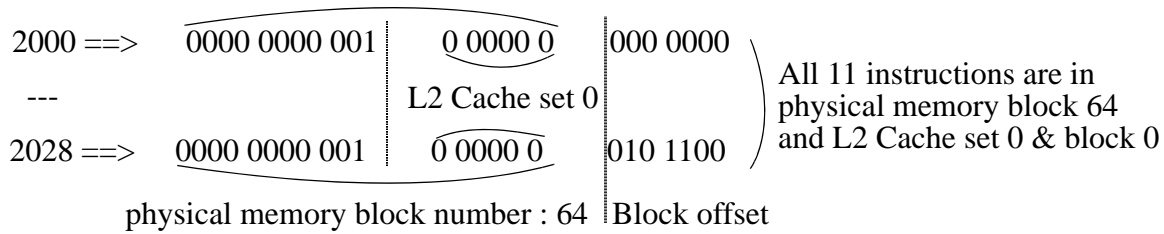
A7) The number of cache blocks = (cache size in bytes)/(block size in bytes) = $2^{15}/2^7 = 2^8 = 256$

The number of sets in the cache = (number of blocks in cache)/(association degree) = $2^8/2^2 = 2^6 = 64$

The fields of the physical address used by the second level cache :



The address mappings are as follows :



The program generates 31 memory addresses. There are four misses and 27 hits. The hit ratio is 0.87. All four different blocks map to set 0 of the L2 cache. Since the cache is 4-way block-set associative, all four blocks are stored in set 0. There is no need to have any block replacement for the first two iterations.

Address	Physical Memory cache block #	Cache memory set #/block #	Hit/Miss	Block replaced/Block written back
2000	64	0/0	Miss	None
8000	256	0/1	Miss	None
2004	64	0/0	Hit	None
2008	64	0/0	Hit	None
200C	64	0/0	Hit	None
A000	320	0/2	Miss	None
2010	64	0/0	Hit	None
C000	384	0/3	Miss	None
2014	64	0/0	Hit	None
A000	320	0/2	Hit	None
2018	64	0/0	Hit	None
C000	384	0/3	Hit	None
201C	64	0/0	Hit	None
2020	64	0/0	Hit	None
2024	64	0/0	Hit	None
2000	64	0/0	Hit	None
8004	256	0/1	Hit	None
2004	64	0/0	Hit	None
2008	64	0/0	Hit	None
200C	64	0/0	Hit	None
A004	320	0/2	Hit	None
2010	64	0/0	Hit	None
C004	384	0/3	Hit	None
2014	64	0/0	Hit	None
A004	320	0/2	Hit	None
2018	64	0/0	Hit	None
C004	384	0/3	Hit	None
201C	64	0/0	Hit	None
2020	64	0/0	Hit	None
2024	64	0/0	Hit	None
2028	64	0/0	Hit	None

Q8) Consider the following EMY mnemonic machine language program :

```

400000    LW      R10, 0(R8)      # R8 points at array A and initially has 10000000
400004    LW      R11, 0(R9)      # R9 points at array B and initially has 10002000
400008    SLT    R12, R10, R11    # Compare R10 and R11
40000C    BEQ    R12, R0, 1       # Skip next instruction if R10 is not less than R11
400010    SW      R11, 0(R8)      # Store the 2nd number in the first
400014    ADDI   R8, R8, 4         # Update the array A pointer
400018    ADDI   R9, R9, 4         # Update the array B pointer
40001C    ADDI   R13, R13, (-1)10 # The loop-end counter is R13
400020    BNE    R13, R0, (-9)10 # If not the end, go back to 400000

```

Assume that the EMY CPU is **unpipelined**, as discussed in **Chapter 4** of the textbook. Assume also that the EMY computer has a **memory hierarchy** now. That is, **all** the addresses in the program above are **virtual addresses**. Finally, assume that the BEQ instruction in location 40000C is **untaken** for **all** iterations of the loop : SW is executed in every iteration.

a) Assume that the hit time is 1 clock period and the average memory stall cycles per instruction is 3.58. Assume also that there are **1000** iterations of the loop, i.e. R13 is $(1000)_{10}$, and the miss penalty is 50 clock periods. Calculate the miss ratio.

b) Assume that the EMY computer has physical Level 1 instruction and data caches each one of which is a 16KB cache with direct mapping and with write-back and write allocate. The block length is 16 bytes. The physical memory contains 16MBytes. The page size is 4096 bytes.

Assume that **R13 is 4 initially**. In addition, assume that

- „ The instructions are in physical memory locations 5000 through 5020,
- „ Data elements that start at virtual address 10000000 are in 6000 through 600C, and
- „ Data elements that start at virtual address 10002000 are in 7000 through 700C.

Clearly show which instruction/data are in which cache, which cache block and which physical memory block for **all** instructions and data as done in class.

c) Assume again that **R13 is 4 initially**. The physical memory is **4-way** low-order interleaved. Show how the cache blocks in part (b) are mapped to the memory banks : Show which instruction and data are stored in which location of which memory bank for **all** instructions and data as done in class.

d) Assume again that the page size is 4096 bytes. Clearly show the **virtual** and **physical** page numbers of all instructions and data if **R13 = 4**.

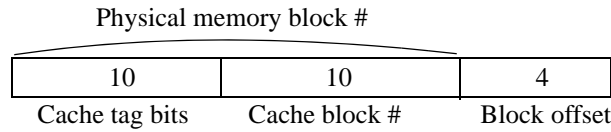
A8) a) In order to calculate the miss ratio, first we need to obtain the total number of memory accesses for the program. Its calculation is the same as the previous question : It is the sum of the number of instruction fetches and the number of data accesses. LW and SW instructions require two memory accesses each. The other instructions require one memory access. There are nine instructions in the loop and we execute all of them : four A/L, two control and three L/S. For a single iteration, we have then 12 memory accesses.

For 1000 iterations, there are 12000 memory accesses. There are 1000 iterations, therefore, 9000 instructions are executed.

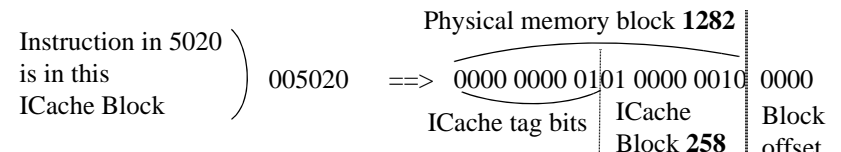
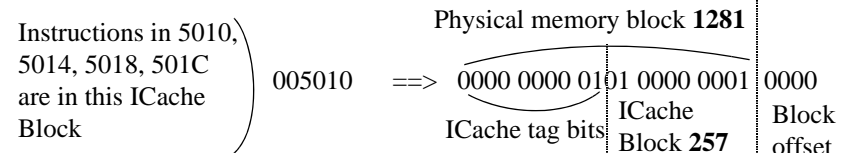
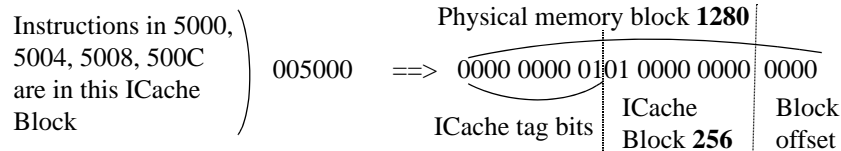
Average memory accesses per instruction is the number of memory accesses for the program/NI = $12000/9000 = 1.33$.

The average memory stall cycles per instruction is the average memory accesses per instruction * miss ratio * miss penalty : $3.58 = 1.33 * \text{miss ratio} * 50$. Therefore, miss ratio = $3.58/(1.33*50) = \mathbf{0.0538}$ which is **5.38%**.

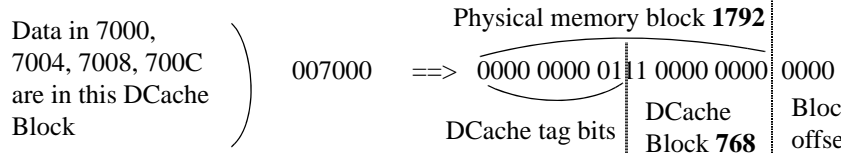
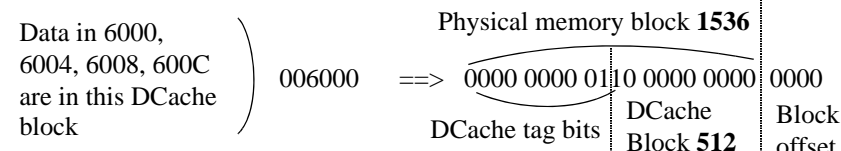
b) The physical memory has 16MB which is 2^{24} bytes. Therefore, the physical address has **24** bits. The physical memory (PM) has (PM size in bytes)/(block size in bytes) $2^{24}/2^4 = 2^{20}$ blocks. The cache memories have 16KB each which is 2^{14} bytes. The number of cache blocks = (cache size in bytes)/(block size in bytes) = $2^{14}/2^4 = 2^{10} = 1024$ blocks in each cache. The physical address from the point of physical memory and cache memory blocks is as follows :



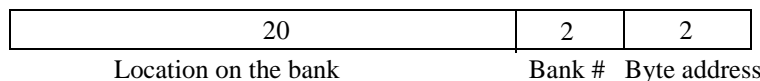
The instruction cache, the ICache :



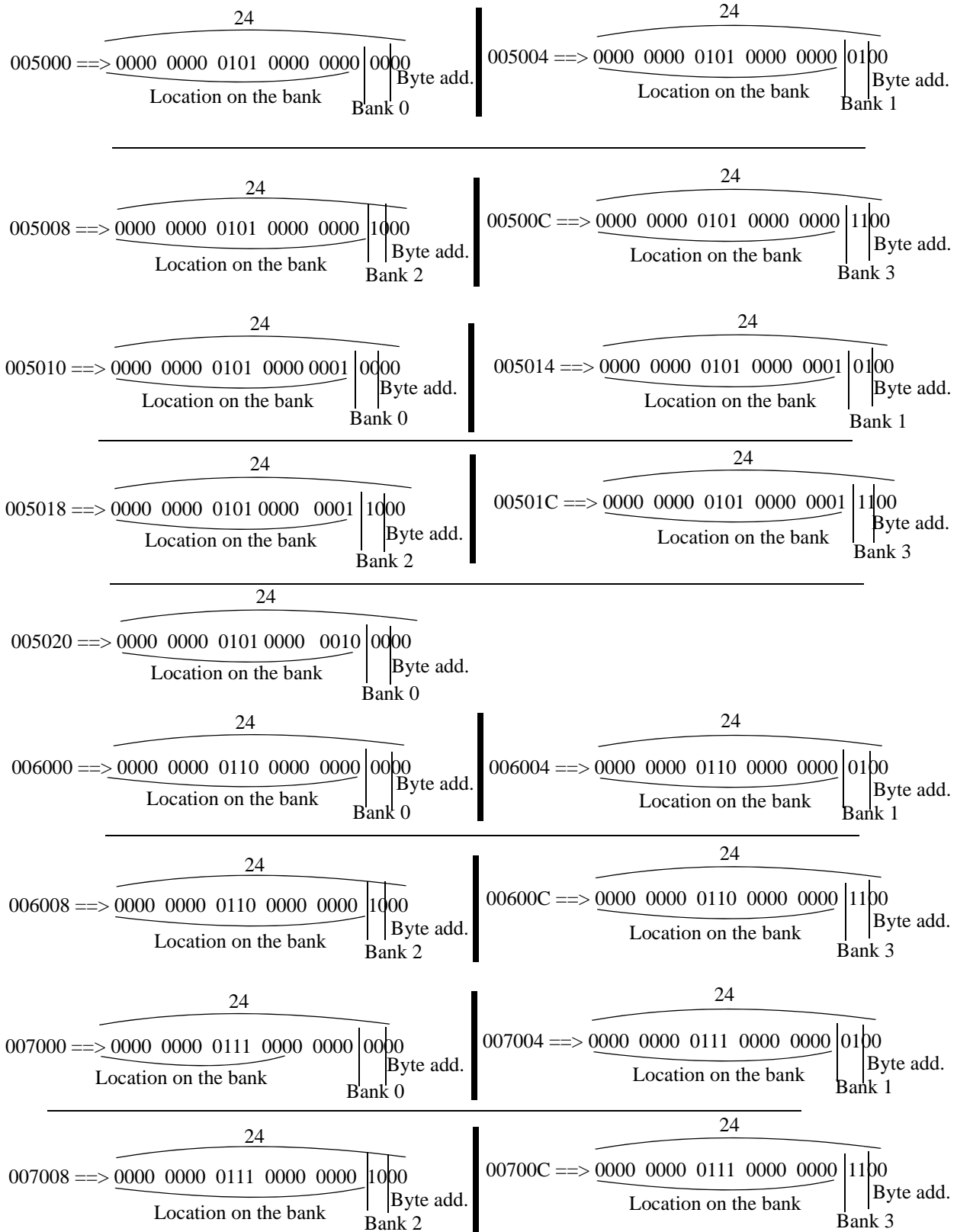
The data cache, the DCache :



c) 4-way interleaving requires two bits for the bank number. The physical memory address fields are as follows :



The mapping of the instructions and data of the above program to the physical memory banks is as follows :



The memory banks are as follows :

Bank #	0	1	2	3
Physical addresses Their content	0 ?	4 ?	8?	C ?
Physical addresses Their content
Physical addresses Their content	5000 LW R10, 0(R8)	5004 LW R11, 0(R9)	5008 SLT R12, R10, R11	500C BEQ R12, R0, 1
Physical addresses Their content	5010 SW R11, 0(R8)	5014 ADD R8, R8, 4	5018 ADDI R9, R9, 4	501C ADDI R13, R13, (-1) ₁₀
Physical addresses Their content	5020 BNE R13, R0, (-9) ₁₀			
Physical addresses Their content
Physical addresses Their content	6000 Array A element	6004 Array A element	6008 Array A element	600C Array A element
Physical addresses Their content
Physical addresses Their content	7000 Array B element	7004 Array B element	7008 Array B element	700C Array B element
Physical addresses Their content
Physical addresses Their content	FFFFE0	FFFFE4	FFFFE8	FFFFEC

d) The virtual page numbers based on the fact that the page length is 4096 bytes = 2^{12} bytes :

$$400000-400020 \implies \underbrace{0000\ 0000\ 0100\ 0000\ 0000}_{\text{Virtual page number 1024}} \mid \underbrace{0000\ 0000\ 0000}_{\text{Page offset}}$$

$$10000000-1000000C \implies \underbrace{0001\ 0000\ 0000\ 0000\ 0000}_{\text{Virtual page number 65536}} \mid \underbrace{0000\ 0000\ 0000}_{\text{Page offset}}$$

$$10002000-1000200C \implies \underbrace{0001\ 0000\ 0000\ 0000\ 0010}_{\text{Virtual page number 65538}} \mid \underbrace{0000\ 0000\ 0000}_{\text{Page offset}}$$

The physical page numbers based on the fact that the page length is 4096 bytes = 2^{12} bytes :

5000-5020 ==> 0000 0000 0101 | 0000 0000 0000
 Physical page number 5 | Page offset

6000-500C ==> 0000 0000 0110 | 0000 0000 0000
 Physical page number 6 | Page offset

7000-7020 ==> 0000 0000 0111 | 0000 0000 0000
 Physical page number 7 | Page offset

Q9) Consider the following piece of EMY mnemonic machine language program :

```

400000    ADDI    R8, R0, 0
400004    LW     R9, 0(R10)      # R10 points at array A which starts at 10000000
400008    ADD    R8, R8, R9
40000C    ADDI   R10, R10, 4
400010    ADDI   R11, R11, (-1)10 # R11 is the loop-end counter
400014    BNE   R11, R0, (-5)10
400018    SW    R8, 0(R10)
  
```

Assume that the EMY CPU is **unpipelined**, as discussed in **Chapter 4** of the textbook. Assume also that the EMY computer has a **memory hierarchy** now. That is, **all** the addresses in the program above are **virtual addresses**.

Assume that the EMY computer has physical Level 1 instruction and data caches each one of which is a 4KB cache with direct mapping and with write-back and write allocate. The block length is 16 bytes. The physical memory contains 16MBytes. The page size is 4096 bytes.

a) Assume that **R11 is 2 initially**. In addition, assume that

- The instructions are in physical memory locations 2000 through 2018,
- Data elements are in physical memory locations 5000 through 5008

i) **Clearly** show which instruction/data are in which cache, which cache block and which physical memory block for **all** instructions and data as done in class.

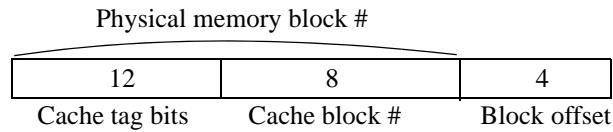
ii) Then, assume that it is a **cold** start and, indicate chronologically if there is a cache miss or hit for all memory accesses generated :

Address	Physical memory block #	Which cache and which block	Hit/Miss
2000	512	Instruction Cache block 0	Miss
...

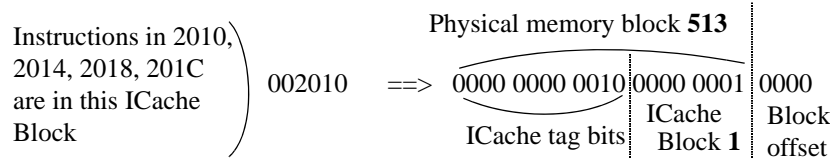
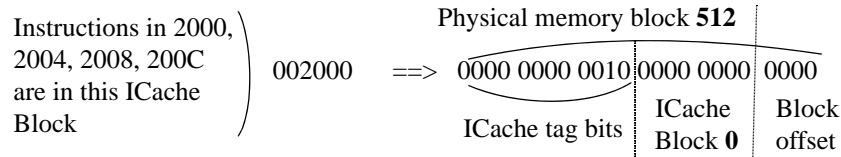
b) Assume now that there are **(1000)₁₀** iterations. Calculate the **miss ratio**. Show your work.

Assume that there will not be any TLB misses nor page faults during the execution.

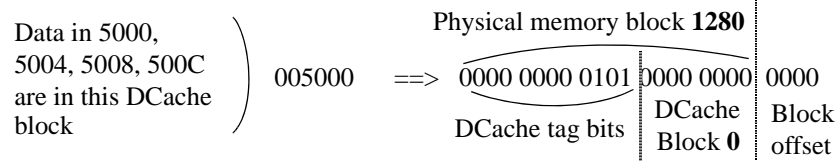
A9) a) The physical memory (PM) has $16\text{MB} = 2^{24}$ bytes. Then, the physical address has **24** bits. The physical memory has $(\text{PM size in bytes})/(\text{block size in bytes}) = 2^{24}/2^4 = 2^{20}$ blocks. The cache memories have 4KB each which is 2^{12} bytes. The number of cache blocks = $(\text{cache size in bytes})/(\text{block size in bytes}) = 2^{12}/2^4 = 2^8 = 256$ blocks in each cache. The physical address from the point of physical memory and cache memory blocks is as follows :



i) The instruction cache, the ICache :



The data cache, the DCache :



ii) We continue with the table as shown below :

Address	Physical memory block #	Which cache and which block	Hit/ Miss
2000	512	ICache block 0	Miss
2004	512	ICache block 0	Hit
5000	1280	DCache block 0	Miss
2008	512	ICache block 0	Hit
200C	512	ICache block 0	Hit
2010	513	ICache block 1	Miss
2014	513	ICache block 1	Hit

Address	Physical memory block #	Which cache and which block	Hit/ Miss
2004	512	ICache block 0	Hit
5004	1280	DCache block 0	Hit
2008	512	ICache block 0	Hit
200C	512	ICache block 0	Hit
2010	513	ICache block 1	Hit
2014	513	ICache block 1	Hit
2018	513	ICache block 1	Hit
5008	1280	DCache block 0	Hit

b) In order to calculate the miss ratio, first we need to know the total number of memory accesses for the program. Its calculation is the same as the previous question : It is the sum of the number of instruction fetches and the number of data accesses. As calculated in the previous question, for 1000 iterations, there are 6003 memory accesses.

We see that we there can be only two misses on the instruction cache for the whole program execution since only two blocks are needed for the program and they do not replace each other again and again.

On the data cache memory side, we see that over 1000 iterations we need to have 1001 memory data accesses for 1001 data elements. 1000 of them by the LW and 1 by the SW. On the data cache, there is a miss every four data elements. Also, data elements do not replace each other again and again. Thus for 1000 iterations, there are 250 misses due to the LW and 1 due to the SW. Therefore, there are 251 misses. The total number of misses is 253.

Therefore, miss ratio = The number of misses/The number of memory accesses = $253/(6003) = 0.042$ which is **4.2%**.

Q10) Consider the following program written for the unpipelined EMY CPU :

```

400C00    LW      R9, 0(R8)           # R8 points at array A and initially has 10EA0000
400C04    ADD     R9, R9, R9
400C08    ADDI    R10, R9, AC
400C0C    SW      R10, 0(R8)
400C10    ADDI    R8, R8, 4
400C14    ADDI    R11, R11, (-1)10    # R11 is the loop-end counter
400C18    BNE     R11, R0, (-7)10

```

Assume that the EMY CPU is **unpipelined**, as discussed in **Chapter 4** of the textbook. Assume also that the EMY computer has a **memory hierarchy** now. That is, **all** the addresses in the program above are **virtual addresses**. The above program is **rewritten** so that all addresses are **physical** addresses :

```

5C00     LW      R9, 0(R8)           # R8 points at array A which starts at 6000
5C04     ADD     R9, R9, R9
5C08     ADDI    R10, R9, AC
5C0C     SW      R10, 0(R8)
5C10     ADDI    R8, R8, 4
5C14     ADDI    R11, R11, (-1)10    # R11 is the loop-end counter
5C18     BNE     R11, R0, (-7)10

```

Thus,

- The instructions are in physical memory locations **5C00** through **5C18**,
- Data elements are in physical memory locations, starting at **6000**

Assume that

- The EMY computer has physical Level 1 instruction and data caches
 - Each L1 cache is a 8 KB cache with
 - Direct mapping and with write-back and write allocate.
- The block length is 32 bytes.
- The physical memory contains 128MBytes.

Assume that **R11 is 2 initially**.

a) Clearly show which instruction/data are in which cache, which cache block and which physical memory block for **all** instructions and data as done in class.

b) Then, assume that it is a **cold** start and, indicate chronologically if there is a cache miss or hit for all memory accesses generated :

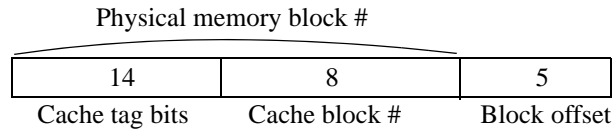
Address	Physical memory block #	Which cache and which block	Hit/Miss
5C00	?	Instruction Cache block ?	?
...

A10) a) The physical memory (PM) has 128MB = 2^{27} bytes. Therefore, the physical address has **27** bits.

The physical memory has (PM size in bytes)/(block size in bytes) $2^{27}/2^5 = 2^{22}$ blocks.

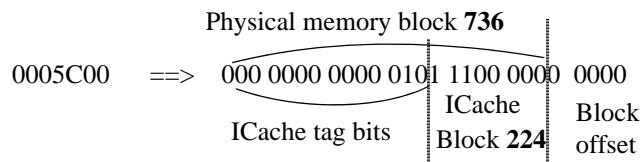
The cache memories have 8KB that is 2^{13} bytes. The number of cache blocks = (cache size in bytes)/(block size in bytes) = $2^{13}/2^5 = 2^8 = 256$ blocks in each cache.

The physical address from the point of physical memory and cache memory blocks is as follows :



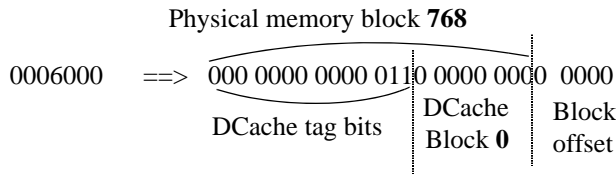
The instruction cache, the ICache :

Instructions in 5C00, 5C04, 5C08, 5C0C, 5C10, 5C14, 5C18, are in this ICache Block



The data cache, the DCache :

Data in 6000, 6004 are in this are in this DCache block



b) We continue with the table as shown below :

Address	Physical memory block #	Which cache and which block	Hit/ Miss
5C00	736	ICache block 224	Miss
6000	768	DCache block 0	Miss
5C04	736	ICache block 224	Hit
5C08	736	ICache block 224	Hit
5C0C	736	ICache block 224	Hit
6000	768	DCache block 0	Hit
5C10	736	ICache block 224	Hit
5C14	736	ICache block 224	Hit
5C18	736	ICache block 224	Hit

Q11) Assume that the EMY computer has a **memory hierarchy**. Consider the following program where all addresses are **physical** addresses :

```

4000    LW      R8, 0(R9)           # R9 points at k which is at 7000
4004    LW      R10, 0(R11)        # R11 points at vector A which starts at 8000
4008    ADD     R12, R8, R10
400C    SW      R12, 0(R13)        # R13 points at vector B which starts at 9000
4010    SUB     R10, R0, R12
4014    SW      R10, 0(R11)
4018    ADDI    R11, R11, 4
401C    ADDI    R13, R13, 4
4020    ADDI    R14, R14, (-1)10   # R14 is the loop-end counter
4024    BNE     R14, R0, (-9)10

```

Thus,

- The instructions are in physical memory locations **4000** through **4024**,
- Data elements are in physical memory locations, starting at **7000**, **8000** and **9000**.

Assume that

- The EMY computer has physical Level 1 instruction and data caches
 - Each L1 cache is a 4 KB cache with
 - Direct mapping and with write-back and write allocate.
 - The block length is 16 bytes.
 - The physical memory contains 256MBytes.

a) Assume that **R14** is **2** initially. **Clearly** show which instruction/data are in which cache, which cache block and which physical memory block for **all** instructions and data as done **in class**.

b) Assume that the EMY CPU is **unpipelined**, as discussed in class. Based on your answer to part (a) above and assuming that it is a **cold** start, indicate chronologically if there is a cache miss or hit, etc. for all memory accesses generated :

Address	Physical memory block #	Which cache and which block	Hit/Miss	Cache Read or Write	Block replaced/Block written back
4000	?	Instruction Cache block ?	?	Read	?
...		

c) Assume for this part that the EMY CPU is **pipelined** as discussed in class. Assume that the L1 cache memories take one (1) clock period when there is a hit. When there is a miss, the latency is four (4) clock periods and transferring a 4-byte item takes 1 clock period each.

i) Reorder the instructions, without increasing the number of instructions in the above code so that it can run efficiently and correctly on the pipelined EMY CPU as shown **in class**.

ii) Based on your answers to parts (a) and (b), clearly show in which clock period the last cycle of the last instruction is completed as done in class, i.e. together with all the necessary forwarding, register-reading-writing and resolved data hazard types as done in class.

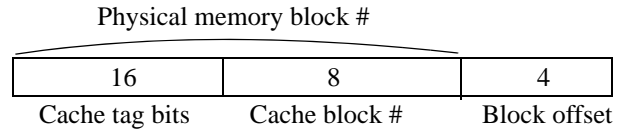
A11) a) The physical memory (PM) has $256\text{MB} = 2^{28}$ bytes. Therefore, the physical address has **28** bits.

The physical memory has $(\text{PM size in bytes})/(\text{block size in bytes}) = 2^{28}/2^4 = 2^{24}$ blocks.

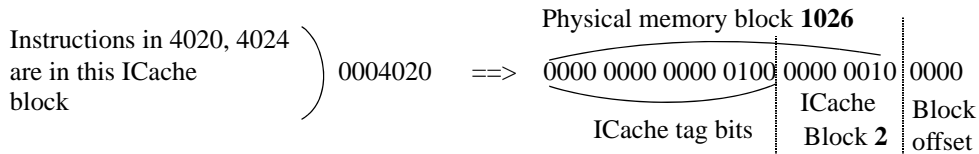
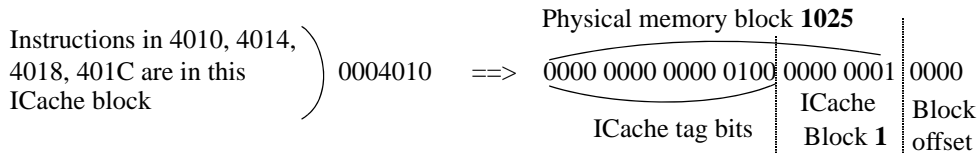
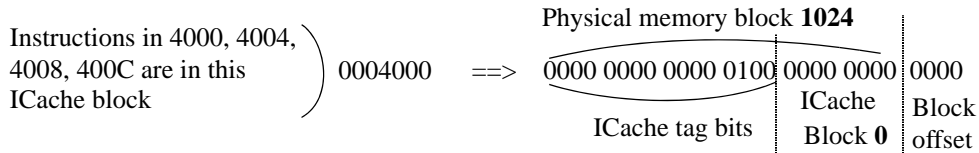
The cache memories have 4KB that is 2^{12} bytes.

The number of cache blocks = $(\text{cache size in bytes})/(\text{block size in bytes}) = 2^{12}/2^4 = 2^8 = 256$ blocks in each cache.

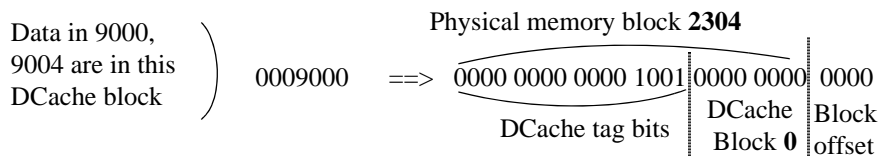
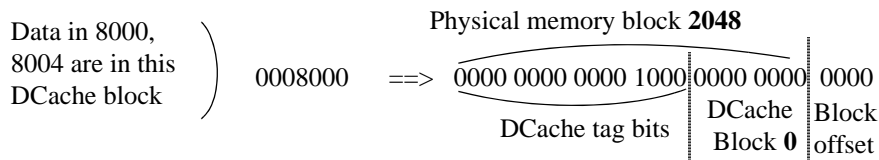
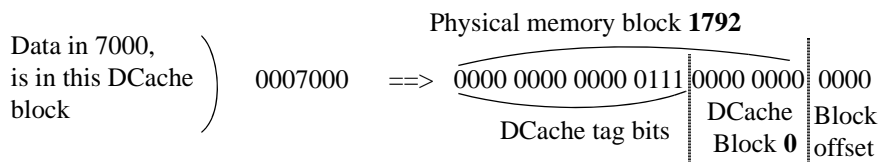
The physical address from the point of physical memory and cache memory blocks is as follows :



The instruction cache, the ICache :



The data cache, the DCache :



b) We continue with the table as shown below :

Address	Physical memory block #	Which cache & which block	Hit/ Miss	Cache Read or Write	Block replaced/ written back
4000	1024	ICache block 0	Miss	Read	None
7000	1792	DCache block 0	Miss	Read	None
4004	1024	ICache block 0	Hit	Read	None
8000	2048	DCache block 0	Miss	Read	PM block 1792 overwritten
4008	1024	ICache block 0	Hit	Read	None
400C	1024	ICache block 0	Hit	Read	None
9000	2304	DCache block 0	Miss	Write	PM block 2048 overwritten
4010	1025	ICache block 1	Miss	Read	None
4014	1025	ICache block 1	Hit	Read	None
8000	2048	DCache block 0	Miss	Write	PM block 2304 written back
4018	1025	ICache block 1	Hit	Read	None
401C	1025	ICache block 1	Hit	Read	None
4020	1026	ICache block 2	Miss	Read	None
4024	1026	ICache block 2	Hit	Read	None
4004	1024	ICache block 0	Hit	Read	None
8004	2048	DCache block 0	Hit	Read	None
4008	1024	ICache block 0	Hit	Read	None
400C	1024	ICache block 0	Hit	Read	None
9004	2304	DCache block 0	Miss	Write	PM block 2048 written back
4010	1025	ICache block 1	Hit	Read	None
4014	1025	ICache block 1	Hit	Read	None
8004	2048	DCache block 0	Miss	Write	PM block 2304 written back
4018	1025	ICache block 1	Hit	Read	None
401C	1025	ICache block 1	Hit	Read	None
4020	1026	ICache block 2	Hit	Read	None
4024	1026	ICache block 2	Hit	Read	None

c) i) The original instructions are reordered to avoid stalls and to fill the branch delay slot :

```

4000    LW      R8, 0(R9)           # R9 points at k which is at 7000
4004    LW      R10, 0(R11)        # R11 points at vector A which starts at 8000
4008    ADDI    R14, R14, (-1)10  # R14 is the loop-end counter
400C    ADD     R12, R8, R10
4010    SUB     R10, R0, R12
4014    SW      R10, 0(R11)
4018    SW      R12, 0(R13)        # R13 points at vector B which starts at 9000
401C    ADDI    R11, R11, 4
4020    BNE    R14, R0, (-8)10
4024    ADDI    R13, R13, 4
    
```

ii) The pipelined execution of the reordered code with cache misses is as follows :

Instruction	IF	ID	EX	MEM	WB	IF	ID	EX	MEM	WB
4000 LW R8, 0(R9)	1/5	6	7	8/12	13					
4004 LW R10, 0(R11)	6	7/12	13	14/18	19 _H	34	35 _H	36	37/41	42 _H
4008 ADDI R14, R14, (-1) ₁₀	7/12	13/18	19	20	21	35	36/41	42	43	44
400C ADD R12, R8, R10	13/18	19 _H	20	21	22	36/41	42 _H	43	44	45 _H
4010 SUB R10, R0, R12	19/23	24	25	26	27	42	43	44	45	46
4014 SW R10, 0(R11)	24	25	26	27		43	44	45	46	
4018 SW R12, 0(R13)	25	26	27	28/32		44	45 _H	46	47/52	
401C ADDI R11, R11, 4	26	27/32	33	34	35 _H	45	46/52	53	54	55
4020 BNE R14, R0, (-8) ₁₀	27/32	33				46/52	53			
4024 ADDI R13, R13, 4	33	33/4	35	36	37	53	54	55	56	57

Above, the cache misses experienced by the LW instruction in the second iteration and SW instruction in location 4018 are due to **conflict** misses where two different physical memory blocks (2048 and 2304) map to the same Data Cache block !

Note that the execution time for an ideal memory case is 23 clock periods !