1. True or False questions. Write T or F in front of the headings. (1X8=8pts) 6

F (1) Given the same block size and total cache size, the hit time of the direct-mapped cache is smaller than the hit time of the fully associative cache.

T (2) Combinational logic always produces the same output if the input does not change.

F (3) Solution to the control hazard is to apply forwarding.

T (4) In 32-bit FP representation, the number 0 is represented by setting both the exponent and the fraction part to all 0s.

F (5) If total cache size remains the same, changing the block size does not affect the miss rate.

T (6) In the fully-associative cache, data from any address can be placed into any cache line.

T (7) In MIPS architecture, you can pass at most 4 arguments using argument registers during a procedure call.

T (8) Instruction streams which are usually highly sequential are more likely to benefit from temporal locality than from spatial locality.

2. Short one sentence questions asking mostly the definition of terminologies. (2X8=16pts)

12

(1) List 5 stages of instruction execution in correct order. (No partial score)

instruction fetch
instruction decode
excute
MEM
writeback

(2) Between two cache write policy, which one may generate data consistency issue in multi-processor and shared cache environment?

write-through

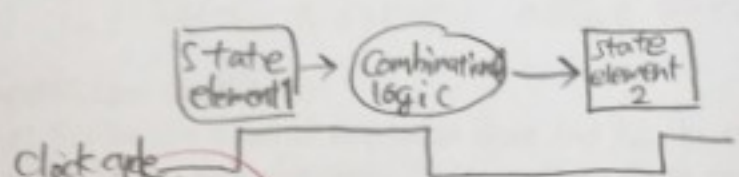
(3) In the pipelined architecture (it is expected that the performance will improve by n times if there are n stages) Give one reason why pipelined datapath design may not reach this theoretical performance limit.

각 단계가 모두 같은 시간에 동시에 끝나지 않는다.
파이프라인 아키텍처에서는 모든 단계가 같은 클럭에 실행되야 하므로
제일 시간 오래 걸리는 단계에 맞추어 cycle 시간을 맞춘다.
따라서 몇몇 단계가 시간내에 다 했지만 쉬게 되는 경우가 생길 수 있으므로
최상의 성능에 도착하지 못

(4) Convert number 0.0000123_{10} into a scientific normalized form in decimal. (No partial score)

1.23×10^{-5}

(5) What does 'edge-triggered clocking' mean?



한 클럭의 마지막에
상태가 바뀐다는 의미다.

(6) For beq instruction why do you need to use 'sign-extend' component in designing the datapath?

beq로 인해 beq instruction의 하위 16비트를 shift left 2한 것과 PC+4가 더해진
branch가 일어나면 결과 값으로 branch하게 된다. 하위 16비트와 PC+4과 더해야
하는데 PC+4의 경우 32 bit이다. 따라서 하위 16비트를 32비트로 부호확장해야

(7) While handling beq instruction, the control unit generate 'branch' signal and do 'AND' with the zero output from the main ALU. Why is 'branch' signal needed?

ALU에서 나온 zero output이 1이라고 해서 무조건 branch가 되면 안된다.
beq와 같은 branch 명령어이고 rs와 rt의 차이가 0일 때만 branch가 일어나야한다
즉 명령어가 beq인지 아닌지 판단하기 위해 branch signal이 필요하다.

(8) What is the name of the digital logic component that selects from n inputs only one signal as an output?

멀티플렉서

3. Explain the write-back and write-through policy in words. No drawing/figure/diagram allowed. (4pts)

write-back은 cache에 유효한 값이 있지않으면 chache에 쓰고. 메모리에 쓰지 않고 있다가 다른 데이터가 그 블록에 들어가려고 할 때 그 때 메모리에 값을 쓰고 다음 데이터를 캐시에 쓰는 방식이다.

write-through은 cache에 데이터를 쓴 메모리에도 쓰므로서 cache와 메모리의 데이터를 동기화하는 방법이다.

4. Pipeline Hazards. (3+3+3=9pts)

(a) List three hazard types.

1. structural hazard
2. data hazard
3. control hazard

3

(b) For each hazard, explain why it happens.

1. 하드웨어가 지원하지않아서 생긴다. 예를 들어 구조적으로 메모리가 하나밖에 못쓰는데 서로 다른 단계에서 쓰려고 할 때 생긴다.
2. 다음 단계 혹은 다다음 명령어가 현재 write하려는 register에 값을 쓰기도 전에 그 레지스터를 읽어와서 생긴다.
3. beq에서 branch할 때 생긴다. ALU단계에 가서 그 때 branch할지 안할지 결정할수 있는데 그 때는 이미 그 다음 명령어들이 파이프라인 프로세서에 들어와 있다.

3

(c) List solutions to three hazards, at least one solution each. Simply stalling is not considered a solution.

1. 하드웨어를 추가한다
2. 데이터 hazard가 일어날수 있는지를 WB단계 가기전에 미리 확인해서 포워딩을 한다.
3. 하드웨어를 추가하거나 예측을 한다.

3

5. Amdahl's Law. (4+4=8pts)

(a) Let T_{old} be the original execution time and T_{new} be the improved execution time of a program after applying the enhancement. There are two parts you can improve. Part A can be improved k times and the proportion of Part A in the program is α, (0≤α≤1). Part B can be improved m times and the proportion of Part B is β, (0≤β≤1). Write an equation for the overall speed up S in terms of α, β, k and m.

$$\frac{1}{1-\alpha-\beta+\frac{\alpha}{k}+\frac{\beta}{m}}$$

0

(b) Your program uses integer arithmetic as well as floating point arithmetic. Integer instruction are 20% and FP instructions are 30%. You have found a way to improve the integer arithmetic instructions 1.2 times faster. If you want to make the overall performance speed-up of 1.2, how much speed-up do you need from the FP instructions?

0

$$\frac{3}{10}\times 1 + \frac{2}{10}\times\frac{12}{10} + \frac{5}{10} = \frac{3}{10}\times 1 + \frac{24}{100} + \frac{1}{2}$$

2/

cf: 1.8

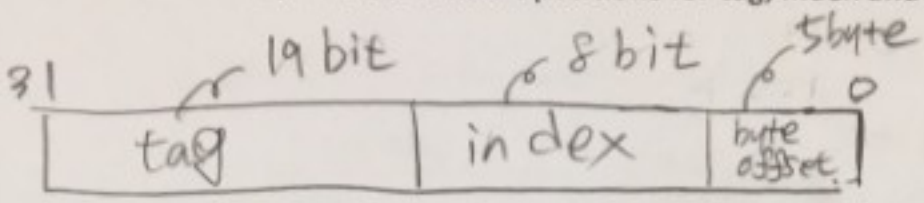
$$\frac{\frac{3}{10}\times 1 + \frac{74}{100}}{} = \frac{12}{} \Rightarrow \frac{3}{10}\times 1 = \frac{46}{100}$$

6. Set associative cache. (3+3=6pts)

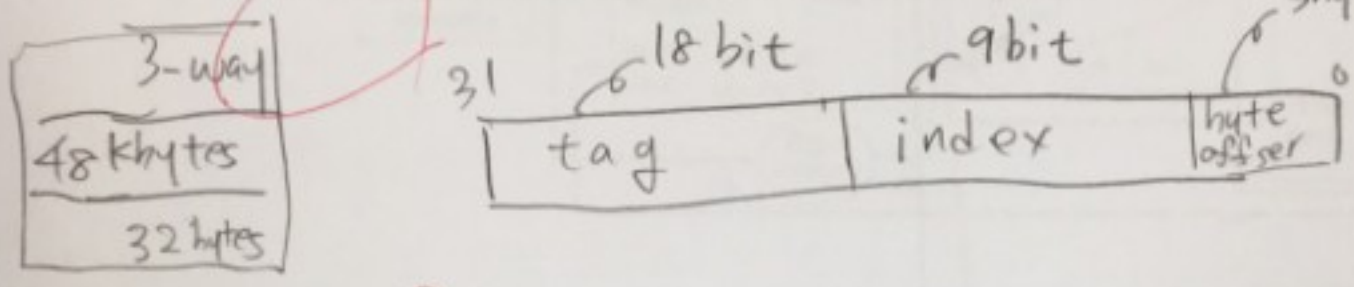
Let's assume we have a cache with this property.

Property	Value
Associativity	4-way
Cache size	32 Kbytes
Block size	32 bytes

(a) For a given 32-bit address, what are the bit positions for tag, index and byte-offset?



(b) If you increase the cache size by 50% and reduce the associativity to 3-way, what are the bit positions for tag, index and byte-offset within a 32-bit address?



7. Following information is given about a CPU cache. (3+4=7pts)

- Instruction cache miss rate: 0.2%
- Data cache miss rate: 0.5%
- Miss penalty: 200 cycles
- Load and store: 20% of all instructions
- CPU has L1 instruction cache and L1 data cache. There are no L2 or L3 cache.
- Ideal case CPI = 2

(a) How much faster is the CPU with a perfect cache (0% miss rate) than this one above?

$I \times \frac{2}{1000} \times 200 = 0.4I$

$I \times \frac{20}{100} \times \frac{5}{1000} \times 200 = 0.2I$

CPU cycles + Memory stall cycles = $I + 0.4I + 0.2I = 1.6I$

$\frac{1.6I}{I} = 1.6$

perfect cache 가 위에거보다 1.6 배 빠르다

(b) You have found a way to reduce the miss rate of either the instruction cache or data cache to half of current miss rate, but not both. Between instruction and data cache, which one would you choose to reduce the miss rate of? Why?

Instruction cache miss rate를 줄일 것이다.

Instruction cache miss rate 가 절반이 된다면 0.1 % 이고

$I \times \frac{1}{1000} \times 200 = 0.2I$ 이므로 CPU cycles + Memory stall cycles = 1.4I 이고

$\frac{1.6I}{1.4I} = \frac{8}{7}$ 이므로 기존보다 $\frac{8}{7}$ 배 성능이 좋다.

반면에 Data cache miss rate가 절반이 된다면 0.25 % 이고

$I \times \frac{20}{100} \times \frac{25}{10000} \times 200 = 0.1I$ 이므로 CPU cycles + Memory stall cycles = 1.5I 이고

$\frac{1.6I}{1.5I} = \frac{16}{15}$ 이므로 기존보다 $\frac{16}{15}$ 배 성능이 좋다.

1.5

9. Operation of Cache. (7+3=10pts)

We have L1 and L2 cache system with these specifications.

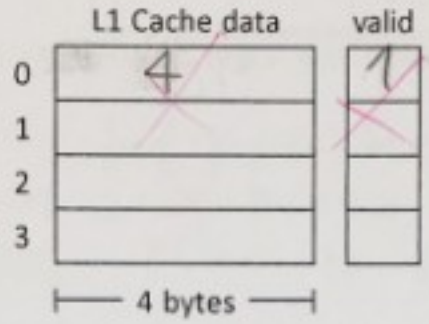
	L1	L2
Associativity	1-way	2-way
Cache size	16 bytes	32 bytes
Block size	4 bytes	8 bytes
Write policy	write-back	write-through

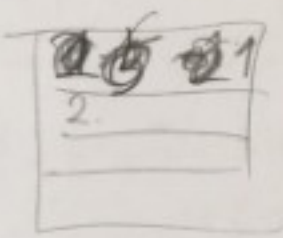
Eviction policy is to choose the oldest one among candidates.
All memory location is initialized to 0.

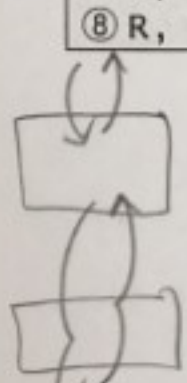
(a) Fill in the contents (in decimal) of both L1 and L2 cache after executing these 8 instructions. First column is Read/Write, second column the byte address and the last column the data value. Tag field is omitted for simplicity. But, you should remember which address maps to which entry in the cache to be able to answer the question.

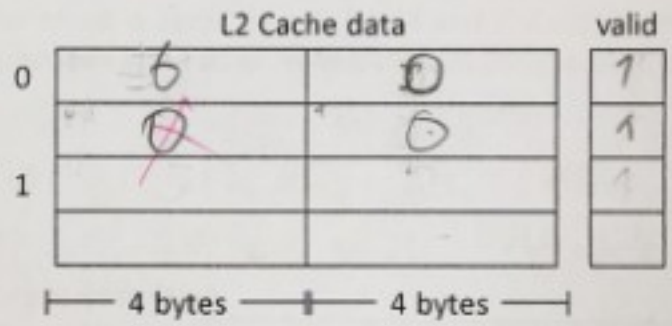
Memory requests

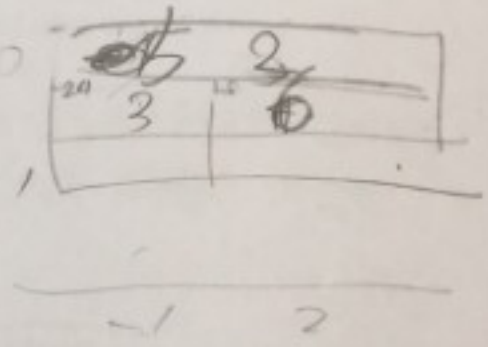
① W, 16, 1
② W, 20, 2
③ W, 16, -1
④ W, 24, 3
⑤ W, 28, 4
⑥ W, 32, 5
⑦ W, 28, 6
⑧ R, 4











(b) L1 access latency is 4 cycles. L2 access latency is 10 cycles. Memory access latency is 140 cycles. What is the average access latency (in cycles) per instruction after executing first 5 instructions from above?

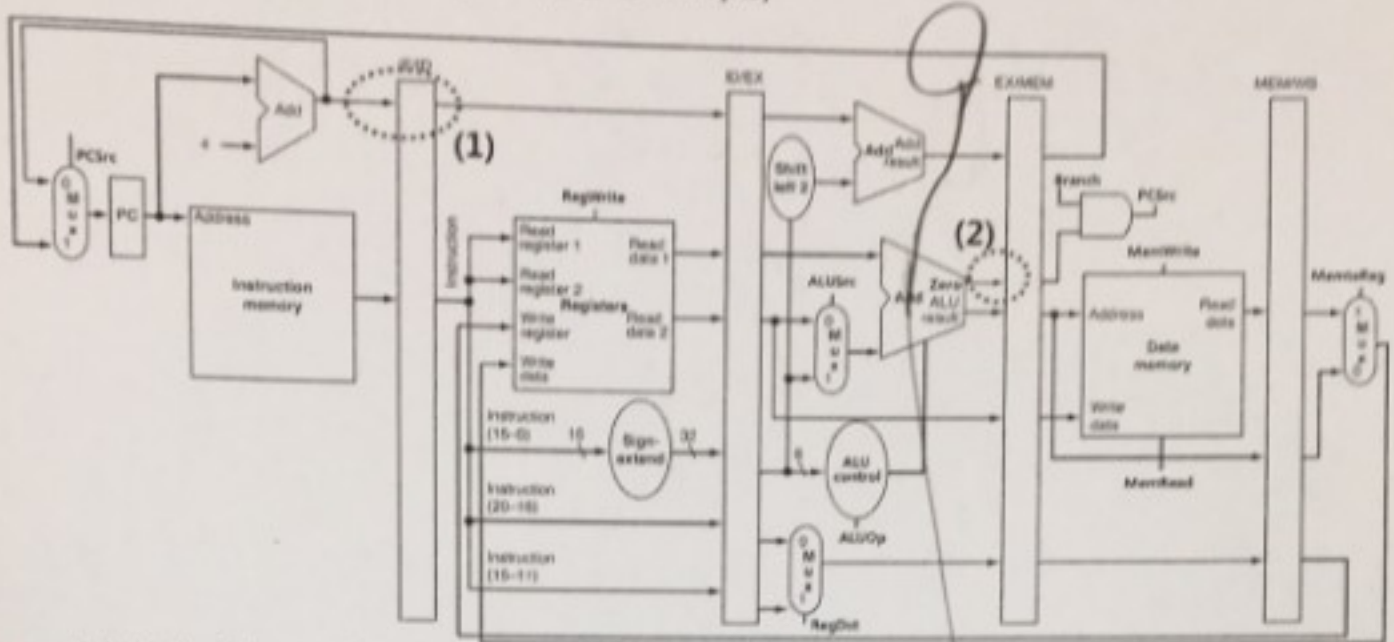
1.5

① 4
② 4+10+140
③ 4
④ 4+10+140
⑤ 4+10+140
⑥ 4+10+140
⑦ 4+10+140
⑧ 4+10+140

avg: 4

10

10. Understanding pipeline datapath. (2+2+6+3=13pts)



2 (a) In the diagram above, there is a region with **Label (1)**. Why do we need to save PC+4 to IF/ID pipeline register?

beq 명령어의 경우 나중에 branch 해야 할 주소를 계산해야 하는데 그 때 PC+4가 쓰인다. 그러니 나중에 EX 단계에서 beq의 PC+4가 필요하므로 PC+4 to IF/ID에 save한다.

(b) For beq instruction, the result of comparing two registers is available in EX stage as shown at **Label (2)**. But, the result is saved into EX/MEM and the action of updating the PC to the branch address is done in MEM stage. Why is this not done in the EX stage?

2 EX stage에서 할 경우 하드웨어를 더 추가해야한다 그러면 EX의 시간이 증가하게 되고 모든 단계의 clock cycle 시간이 늘어나서 성능이 안 좋아질 수 있기 때문이다.

(c) The JALR is an R-type instruction.

JALR \$rs, \$rd	op	rs	rt	rd	shamt	funct

3 It saves the next instruction address to \$rd and jumps to the address stored in \$rs. Explain how to modify the diagram to support this JALR instruction. Be sure to describe what to do in which stage.

rd로 값저장하는 WB 부분을 WB까지 끌고 가야 필요하다.

ID 단계에서 Write data에 PC+4도 주고 그러면 입력이 2개이므로 mux도 달아준다. 또한 그 mux를 제어할 신호를 WB까지 끌고간 후 나중에 WB가 실행될 때 그 mux에 줌으로써 mux의 출력값을 제어한다.
EX 단계에서 rs 값을 끌고와서 $의 부분에 입력값으로 주고 mux도 달아준다. control은 opcode를 보고 알맞는 ALUOP를 ALU control에 주고 ALU control은 funct 코드를 보고 ALU control 신호를 그 mux에게 준다.

(d) How many instructions after JALR needs to be flushed? Why?

3 2번 flush가 필요하다.
branch할 때 EX 단계에서 알 수 있으므로 이미 IF와 ID에 다른 두 명령어가 들어와 있는 상태다. 따라서 2번 flush가 필요하다.