University of California, Berkeley – College of Engineering

Department of Electrical Engineering and Computer Sciences

Fall 2017 Instructors: Randy Katz, Krste Asanovic 2017-10-31

😭 CS61C MIDTERM 2 😃

After the exam, indicate on the line above where you fall in the emotion spectrum between "sad" & "smiley"...

Last Name	Perfect
First Name	Peter
Student ID Number	0xFFFFFFFF
CS61C Login	**cs61c-ZZZ**
*The name of your **SECTION** TA and time*	
Name of the person to your LEFT	
Name of the person to your RIGHT	
*All the work is my own. I had no prior knowledge of the exam contents nor will I share the contents with others in CS61C who have not taken it yet. **(please sign)***	*Peter Perfect*

Instructions (Read Me!)

- This booklet contains 12 numbered pages including the cover page and an answer sheet.
- Please turn off all cell phones, smartwatches, and other mobile devices. Remove all hats & headphones. Place your backpacks, laptops and jackets under your seat.
- You have 80 minutes to complete this exam. The exam is closed book; no computers, phones, or calculators are allowed. You may use one handwritten 8.5"x11" page (front and back) crib sheet in addition to the RISC-V Green Sheet, which we will provide.
- **Please write all of your answers on the answer sheet provided. Only the answer sheet will be graded.** When we provide a blank, please fit your answer within the space provided. **When we provide a multiple choice question, please bubble in your choice(s) on the answer sheet. You will lose credit if you fail to do so.**

	Q1	Q2	Q3	Q4	Q5	Total
Points Possible						90

Q1: RISC-Qué Business

Instruction Format

a. As a refresher on **regular** RISC-V instructions, convert the 32-bit instruction below into machine code by filling in the blanks on your answer sheet with the appropriate binary values.

imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode
1111111	01001	10011	000	10010	010 0011

As a Halloween treat, Prof. Asanovic decides to modify the 32-bit RISC-V ISA that we've been using to a compressed version, where **each instruction is 16 bits**. He decides to remove the funct3 and funct7 fields completely, but to keep the opcode field with the same number of bits as before. **All other specifications remain the same as normally seen in class with 32-bit RISC-V.** Consider this modified 16 bit ISA for the remainder of the problem. Remember to **bubble** in your answers for the following questions on the answer sheet.

b. Considering **solely** R-type instructions, if we now only have an opcode field and register fields, what is the maximum number of registers that could be supported by our compressed instruction format?

2 3 4 8 9 12 16 None of the above

16 - 7 = 9 bits. Each field (rs1,rs2, rd) gets 3 bits and can represent 2^3 = 8 registers.

c. Suppose Prof. Asanovic wants to support only 4 registers. How many bits for the immediate field does he have for this modified I-type?

2 3 4 5 6 7 8 9

4 registers require $\log_2(4)$ = 2 bits for rs1 and rd fields. 16 - 2 - 2 - 7 = 5 bits

d. Now, he also modifies jal/UJ-types to no longer have an rd field (x2 is always used as the link register). What is the range of **bytes** that a program can jump to from current PC?

Forward: ___(i)___ bytes from current PC

Backward: ___(ii)___ bytes from current PC

i. 514 512 510 258 256 254 130 128 126

ii. -514 -512 -510 -258 -256 -254 -130 -128 -126

UJ type now has 16-7 = 9 bits for imm field.

The range of bytes it can jump forward is $(2^{\wedge(9-1)} - 1) * 2 = 255*2 = 510$ bytes.

The range of bytes it can jump backward is $(-(2^{\wedge(9-1)}) * 2 = -512$ bytes.

Compiler, Assembler, Linker & Loader

Now we want to build a single assembly file, joak.s, into an executable. The contents of joak.s are below:

```
        lw      t1, 0(sp)
        bne     t1, x0, JOAK
        addi    t1, t1, 0xB00
        jal     TRIX
JOAK:   andi    t1, t1, 0xF00
TRIX:   sw      t1, 0(sp)
```

e. How many passes would the conventional assembler (the assembler presented in lecture and lab) need to make through this program?

0 1 2 3 4

f. For this specific program, if the assembler now started **backwards** from the last line of the program (labeled TRIX), what is the minimum number of passes that the assembler must make?

0 1 2 3 4

g. For this specific program, which CALL tasks must still be completed?
(select all that apply)

Strip Comments

Expand pseudo-instructions

Generate symbol table

Generate relocation table

Link multiple .out files together

Q2: The Minority Rules

Consider the boolean expression $(\overline{A} + \overline{B})(\overline{AC})(\overline{B} + \overline{C})$. **All answers should be written on your answer sheet.**

a. A partial truth table for two possible combinations of inputs is shown below. Please fill in the missing outputs.

A	B	C	Output
0	1	1	(i) 0
1	0	1	(ii) 0

b. Select all of the following logical expressions that are **equivalent** to the above truth table. **Hint:** There is at least one correct answer, but no more than 4.

$\overline{A}\,\overline{B} + \overline{B}\,\overline{C} + \overline{A}\,\overline{C}$

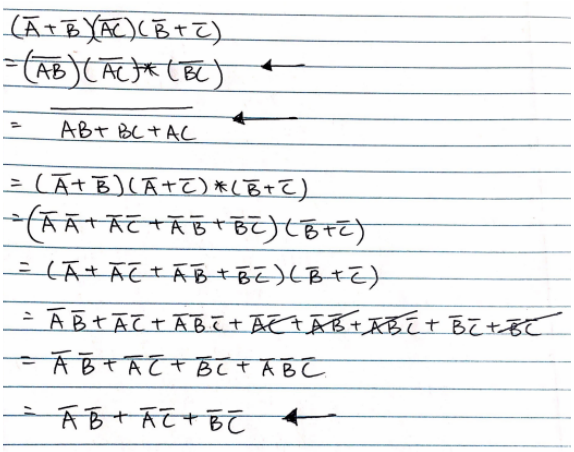
$\overline{AB} + \overline{BC} + \overline{AC}$

$\overline{AB + BC + AC}$

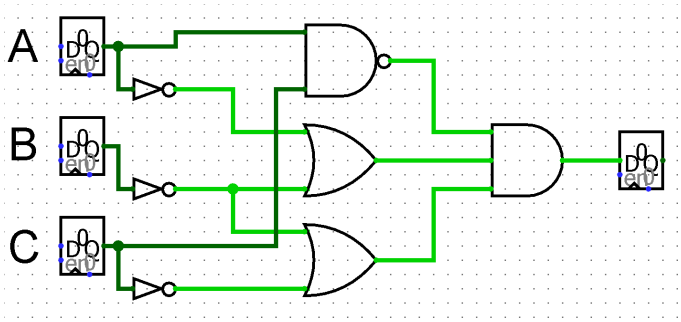
$(\overline{AB})\ (\overline{BC})\ (\overline{AC})$

$\overline{ABC}$

$\overline{A} + \overline{B} + \overline{C}$



c. The circuit below is placed between registers to form a sequential circuit. All three inputs come directly from registers and the output is written straight to a register. Given the setup time is 2ps, hold time is 1ps and clock-to-Q delay is 3ps, along with the propagation delays of each logic gates given below, what is the shortest possible clock period in picoseconds (ps)?

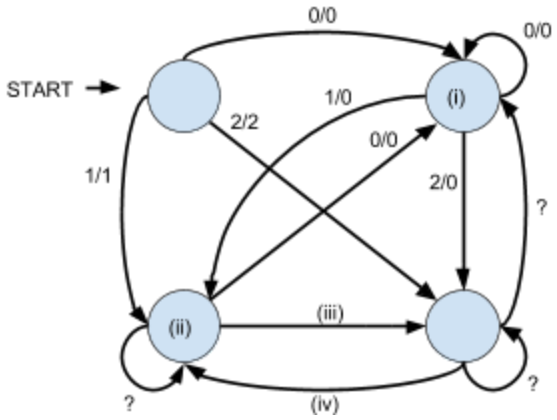


NOT	AND	OR	NAND
1ps	2ps	2.5ps	3ps

8 8.5 9 9.5 10 10.5 11 None of those listed.
Clk-to-q + NOT delay + OR delay + AND delay + setup = 3 + 1 +2.5 + 2 + 2 = 10.5

We now have an FSM that takes in either 0, 1, or 2 as inputs and outputs the minimum value of the last two most recent inputs. For example, an input of 02102122 has the output 00100112. (Note: when the FSM has seen no previous inputs, then the minimum is simply the current input.)

The FSM has already been completed with all necessary arrows, as well as some arrow labels. "?" labels indicate that the label is not provided, but that no question will ask about that label (you may still need to consider what the value of that label should be).



d. What does the state marked with (i) represent?

Most recent input was a 0 Last two inputs were 00

Most recent input was a 1 Last two inputs were 11

Most recent input was a 2 Last two inputs were 12

e. What does the state marked with (ii) represent?

Most recent input was a 0 Last two inputs were 00

Most recent input was a 1 Last two inputs were 11

Most recent input was a 2 Last two inputs were 12

f. What "input/output" pair should the arrow marked with (iii) be labeled with?
0/0 0/1 1/0 1/1 1/2 2/1 2/0

g. What "input/output" pair should the arrow marked with (iv) be labeled with?
0/0 0/1 1/0 1/1 1/2 2/1 2/0

Q3: Trick | Treat, Compare & Set

Consider adding the following instruction to RISC-V:

Instruction	Operation
cse rd, rs1, rs2	if (R[rs1] != R[rs2]) R[rd] = R[rs1] - R[rs2]; else Mem[R[rd]] = 1;

a. What type of instruction will cse be? If multiple formats work, **bubble in** all that apply on your answer sheet.

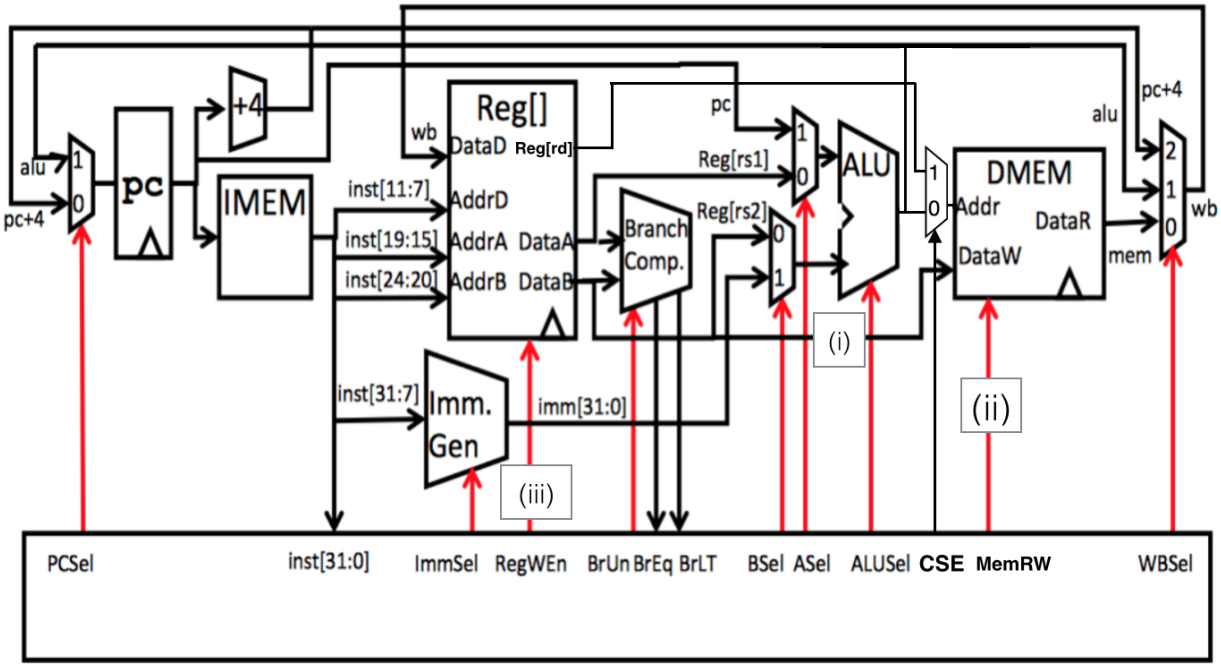
R-type I-type S-type U-type

b. Implement cse in the datapath. Choose all correct implementation for (i), (ii) and (iii), and **bubble** in your answer on your answer sheet.

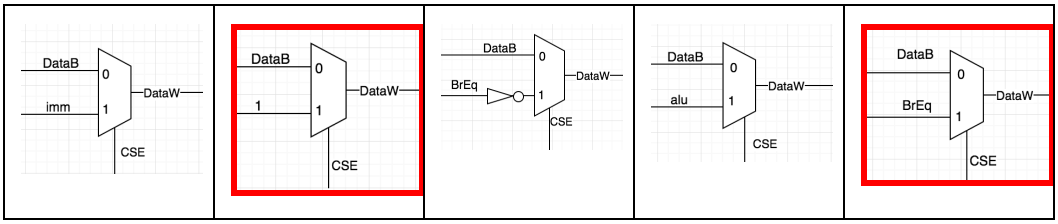
Note 1: The control signal CSE is 1 if and only if the instruction is cse, 0 otherwise.

Note 2: The RegFile in the below datapath has one additional out-port Reg[rd], which outputs the value at AddrD from the RegFile.

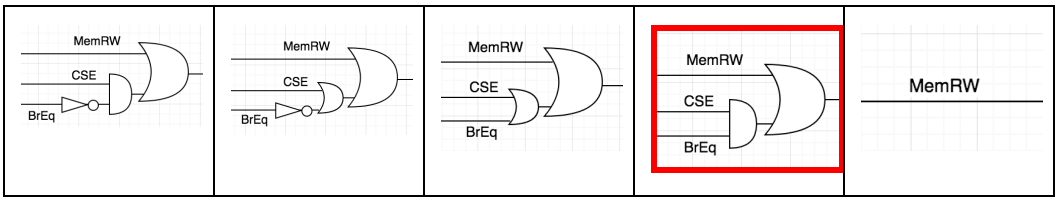
Make sure the functions of the original RISC-V CPU are still preserved!



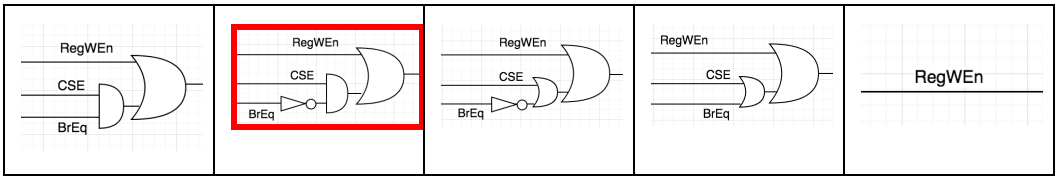
i. Select the correct component to fill in blank (i) in the Execution Stage



ii. Select the correct component to fill in blank (ii) in the Memory Stage



iii. Select the correct component to fill in blank (iii) in the Write Back Stage



c. **Bubble** in the correct control signals for cse on your answer sheet. You may assume the immSel signal is correctly implemented. The possible values for each signal are given below. If the exact value of that signal doesn't matter, then select Don't Care (X). If it is possible for a signal to be "Don't Care", then select "Don't Care" instead of a more specific value (e.g. a value of ⓪ or ① is incorrect when the signal could instead be Ⓧ).

⓪ Signal = 0 ① Signal = 1 ② Signal = 2

Ⓡ Write Disabled Ⓦ Write Enabled Ⓧ Don't Care

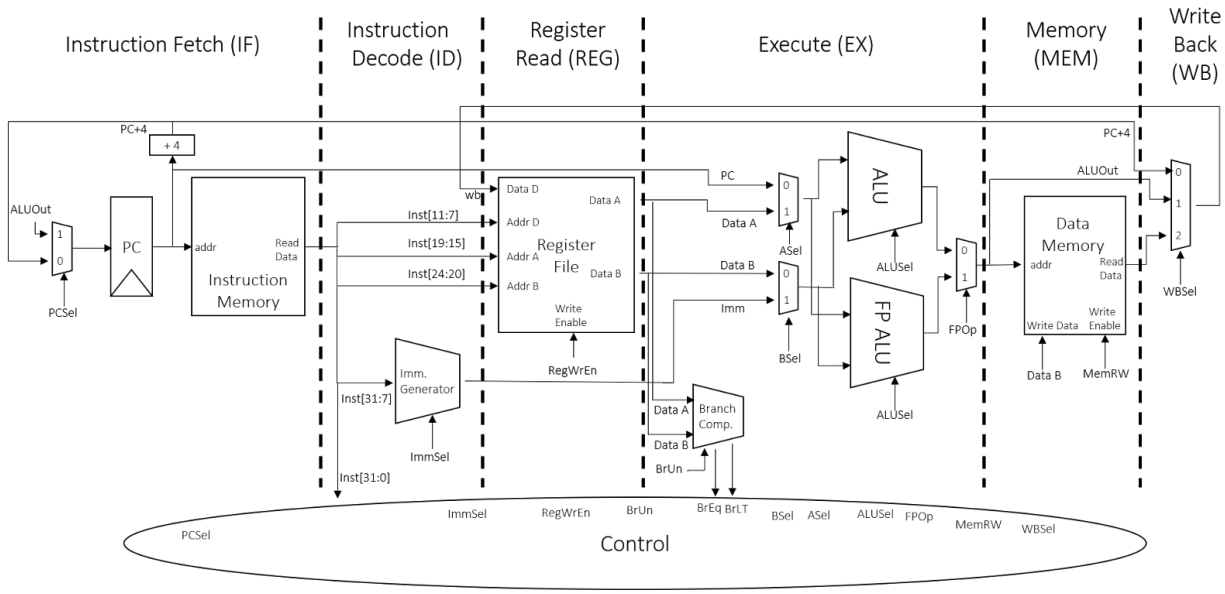
Ⓐ AND Ⓑ OR Ⓒ ADD Ⓓ SUB

CSE	PCSel	RegWEn	BrUn	BSel	ASel	ALUSel	MemRW	WBSel
1	⓪	Ⓡ	Ⓧ	⓪	⓪	Ⓓ	Ⓡ	①

Q4: Danger: Hazardous Material

In this question, you will be working with a modified RISC-V CPU. As opposed to the traditional 5-stage pipeline, this altered CPU has split the second phase into two distinct stages: instruction decode and register read. Furthermore, this CPU has a floating-point ALU (FP ALU) along with a traditional ALU (no floating point Register File is added). The added control signal, FPOp, determines which ALU to use for a given instruction. This floating-point ALU works by interpreting its 32-bit operands in IEEE 754 floating-point format. Unlike in standard RISC-V (and on the green sheet), assume floating point operations use the same registers as normal, non-floating point operations.

A diagram of the modified CPU and its corresponding stages are shown below. To simplify the diagram, any label at the very beginning of a wire acts like a tunnel in Logism. Along with the diagram above, you are given the following delays incurred by the circuit elements in the table below, as well as the delays incurred for some of the datapath stages.



Element	Register CLK-to-Q	Register Setup	MUX	ALU	FP ALU	Mem Read	Mem Write	RegFile Read	RegFile Setup	Imm. Gen.
Delay (ps)	20	25	10	150	215	225	230	100	35	75

a. Consider the floating point instruction, `faddi`, that is similar to `addi` except it considers its operands in floating-point format and executes a floating-point add operation. Assuming that this CPU is **NOT Pipelined** (i.e. it is a single-cycle CPU), what is the shortest clock period possible to execute the instruction `faddi t0, s0, 2.71`

corectly? Write your answer as a single, simplified number (no summations or other expressions) on your answer sheet.

20 + 225 + max(75 + 100) + 10 + 215 + 10 + 10 + 35 = 625 ps

b. What is the shortest possible clock period at which we can run this pipelined CPU? Write your answer as a single, simplified number (no summations or other expressions) on your answer sheet.

IF: CLK-to-Q + MemRead + RegSetup = 20 + 225 + 25 = 270ps
ID: CLK-to-Q + ImmGen + RegSetup = 20 + 75 + 25 = 120ps
REG: CLK-to-Q + RegFileRead + RegSetup = 20 + 100 + 25 = 145ps
EX: CLK-to-Q + MUX + FPALU + MUX + RegSetup = 20 + 10 + 215 + 10 + 25 = 280ps
MEM: CLK-to-Q + MemWrite + RegSetup = 20 + 230 + 25 = 275ps
WB: CLK-to-Q + MUX + RegFileSetup = 20 + 10 + 35 = 65ps
The answer is therefore max(270, 120, 145, 280, 275, 65) = 280ps.

c. Keeping the modified pipeline in mind, consider the program below with the following assumptions:

- **There are no pipelining optimizations** (no forwarding, load delay slot, branch prediction, pipeline flushing, etc.)...
- **We cannot read and write from registers in the same clock cycle.**
- **An integer 100 is stored at memory address 0x61C61C61, and that R[a0] = 0x61C61C61**.
- A hazard between two instructions should be counted as only 1 hazard.

```
lw t0, 0(a0)          # R[a0] = 0x61C61C61
srli s0, t0, 4
faddi s1, t0, 1.7
beq a0, s1, Label
add a1, t2, t3
Label ...
```

Write your answers to the questions below on your answer sheet.

lw t0, 0(a0)
nop
nop
nop
srli s0, t0, 4

```
faddi s1, t0, 1.7
nop
nop
nop
beq a0, s1, Label
nop
nop
nop
add a1, t2, t3
```

i. How many cycles would the program take to execute correctly on the pipelined machine? 19

ii. How many stalls would need to be added for the program to be executed correctly on the pipelined machine? 9

iii. How many data hazards are present in the program? 3

iv. How many control hazards are present in the program? 1

d. For both parts below, reorder the instructions to minimize the number of cycles needed to execute the program, then answer the questions below. Your reordered instructions should produce the same results for all involved registers as the original instructions do.

```
lw t0, 0(a0)
nop
nop
nop
addfi s1, t0, 1.7
srli s0, t0, 4
nop
nop
beq a0, s1, Label
nop
nop
nop
add a1, t2, t3
```

i. How many cycles does the program take to execute with reordered instructions?

18

ii. How many stalls are required?

8

Q5: Do the Monster Cache

a) For the following cache questions, please **bubble** in your answer on the answer sheet:

I. True or False? A fully associative cache has no conflict misses.

II. True or False? A write-back cache must write to memory *immediately* when a block is modified.

For all portions of this question assume that an integer is one word in size and that ALL operations occur from left to right. Consider a 16-way set-associative cache with two-word blocks, 16 sets and a 128 TiB physical byte-addressed address space.

b) When breaking down a physical address into the Tag, Index, and Offset fields, how many bits long is each field? (i.e. what is the T:I:O for the cache?) Write your answer on the blanks provided on your answer sheet.

T: 40 I: 4 O: 3
Total address bits = $\log_2(128Ti) = \log_2(128*2^{\wedge 40}) = 47$
offiset bits = $\log_2(2 \text{ words}) = \log_2(8 \text{ bytes}) = 3$
index = $\log_2(16 \text{ sets}) = 4$
Tag = 47 - 4 - 3 = 40

Now consider the following code segment:

```
void sequence(int* A, int* B) {
    int i;
    //PART C
    for (i = 0; i < 16; i++) {
      B[i] = 2;
      A[i] = 4;
    }
    //PARTS D & E
    for (i = 16; i < 272; i++) {
      B[i] = B[i - 8] + A[i - 8];
      A[i] = B[i - 16] + A[i - 16];
    }
}
```

Let A's address for the following code segment be 0x10000 and B's address be 0x20000. Assume that an integer is one word in size, that ALL OPERATIONS are evaluated from left to right, and that all of the cache's valid bits are set to zero before running the sequence function. Remember to write all of your answers to the questions below on your answer sheet.

c) What is the hit rate for running the loop below **PART C** using the cache from (b)?

1/2 0 7/8 3/4 15/16

Miss/Hit Pattern is MMHH.
The hit rate is 50%.

d) What is the cache hit rate for cache accesses that occur below **PARTS D & E** when running sequence after **PART C** completes.

5/6
The miss/hit pattern is HHMHHM, HHHHHH.
Each 6 accesses are from one iteration of the for loop.
Cache hit rate is 0.5*4/6 + 0.5 * 6/6 = 5/6

e) Assuming the cache and block sizes are held constant, what is the minimum cache associativity that results in a maximum hit rate for the segment of sequence?

1-way 2-way 4-way
8-way 16-way 32-way

The minimum associativity is 2 ways because we have two different arrays A and B and we would to avoid conflict among accesses to them.

f) Assume that sequence ran above resulted in a total of 50 accesses (this may or may not be true) and that it was run on a computer with an L1 and L2 cache. Say that the L1 cache has an access time of 10ns, the L2 cache has an access time of 20ns, main memory has an access time of 50ns, the L1 cache has an 80% hit rate, and that the total AMAT for running sequence is 16 ns. What is the local hit rate for the L2 cache? You do not need to know either of the caches' parameters for this question. **Please write your answer as a decimal, up to 2 decimal places, on your answer sheet.**

10 + 20%(20 + L2localMissrate*50) = 16
10 + 4 + L2localMissrate*10 = 16
L2 local missrate = (16-14)/10 = 0.2
L2 local hit rate = 1- 0.2 = 0.8