CS151B/EE M116C
Computer Systems Architecture
Fall 2017 Sample Midterm Exam I
Instructor: Prof. Lei He

It is a closed-book exam.
There are total NINE pages including this cover page. Check whether you have all pages. If not, let the TA know right now.
Good luck!

Problem 1: ________________ of 10 points
Problem 2: ________________ of 10points
Problem 3: ________________ of 8 points
Problem 4: ________________ of 8 points
Problem 5: ________________ of 12 points
Problem 6: ________________ of 12 points
Total: ________________ of 60 points

Problem 1: (total 10 points)

Short problems:

(1) Name five components of a computer, and give an example for each of them (2 points)

(2) The most important **interface** between hardware and software in the computer architecture is ___________________________________. (1 point)

(3) Target address of conditional branch if it is taken, and target address of jump instruction (3 points. Requirement: specify addresses using bits of instructions)

(4) Among program, compiler, instruction set, computer organization and technology,

___________, ___________ and ___________ can affect CPI. (2 points)

(5) Instruction mix, kernel benchmarks, application benchmarks, and ____________

can be used to evaluate the computer performance. ____________________ among the above is most reliable. (2 points)

Problem 2 (10 points):

Binary bits have no inherent meaning. Given the bit pattern:

0100 0100 0000 1111 1100 0000 0000 0000

What does it represent, assuming that it is

a. An unsigned integer?

b. What is the smallest and largest floating point numbers that can be represented by a 32-bit word according to IEEE standard 754?

c. A MIPS instruction

a) (2 points)

b) (2 points)

c) (3 points)

Now given bit pattern
1001 0100 0000 1111 1100 0100 0000 1000
d. A two's complement integer?

d) (3 points)

Problem 3 (8 points):

For the following C code,

A = B + C
D = A - C

a) write an equivalent pseudo-assembly language program in architectural styles ***memory-memory*** (format: OP M_1 M_2 M_3, where M_i is address of memory) and ***load-store*:**

You may assume that all variables are initially in memory.

(Memory-memory 2 points)

(load-store 3 points)

b) Now assume that OP Code is 6 bits for both architecture, but memory address takes 24 bits in memory-memory instruction, what are the sizes of the codes? And which architecture is more efficient as measured by code size? (1 point)

c) How many memory accesses are there for each code? Which architecture is more efficient as measured by memory bandwidth requirement? (1 point)

d) Assume each arithmetic operation uses 4 cycles, and each memory access uses 10 cycles, how many cycles are needed in total for each code. Which architecture is preferred for high performance and why? (1 point)

Problem 4 (8 points):

A computer designed for Java programs can be speed-up by adding hardware support for garbage collection. Garbage collection currently comprises 24% of the cycles of the workload. There are two possible changes.

(1) Automatically handle garbage collection in hardware. This causes an increase in cycle time by a factor of 1.3. How fast is the program after adding automatically handle garbage collection compared to the original one.

(2) Add new instructions for garbage collection to the ISA. This would halve the number of instructions needed for garbage collection but increase the cycle time by 1.1. How fast is the program after adding automatically handle garbage collection compared to the original one.

(3) Which of the two options, if either, should you choose?

Problem 5 (12 points):

Carry Look-Ahead adder:
Given the following truth table for a full adder:

Inputs			Outputs	
A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

The G (generate) can be defined as $A*B$ and P (propagate) can be defined as $A+B$.

(1) Finish the truth table for C_{out} in a carry-lookahead adder (CLA) where C_{out} is a logic function of G, P and C_{in}, then find the logic function for C_{out} and use Programmable Logic Array (PLA) to implement C_{out} (6 points)

Inputs			Output
G	P	C_{in}	C_{out}
0	0	0	
0	1	0	
1	0	0	-
1	1	0	
0	0	1	
0	1	1	
1	0	1	-
1	1	1	

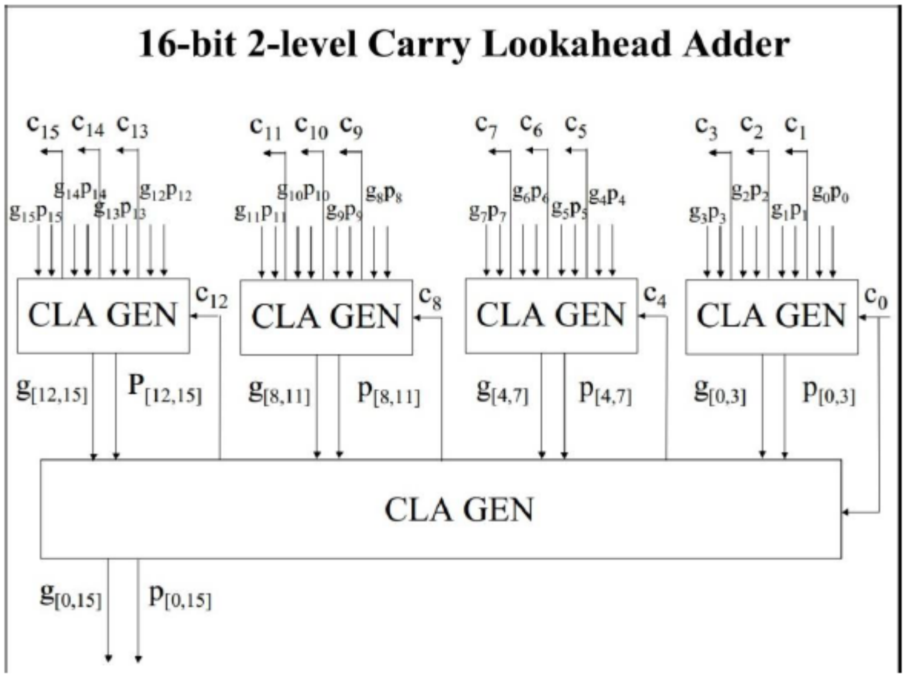
***C**$_{out}$* =
(2 points for equation and 2 points for truth table)

Note: because $G=A*B$ and $P=A+B$, G=0 and P=1 will not occur, so C_{out} should be don't care in these states.

PLA: (2points)

(2) A 16-bit CLA can be built by four 4-bit CLA's in cascade with one CLA GEN, where the carry-out signals from CLA-GEN to the four 4-bit CLA's are C_0, C_4, C_8, and C_{12}. We assume that the inputs to each 4-bit CLA are A_i and B_i (where i is from j + 0 to j + 3, and j is multiple of 4), as well as carry-in C_j, and define $g_i = A_i * B_i$ and $p_i = A_i + B_i$. Then, find logic function of C_4, C_8, C_{12} and $g_{[0, 15]}$ and $p_{[0, 15]}$ using $g_{[n, m]}$, $p_{[n, m]}$, and C_j. (6 points)

(Hint: truth table is not required for the 4-bit CLA once the logic function for 1-bit CLA is given).



C_4 =

C_8 =

$C12$ =

$g[0, 15]$ =

$p[0, 15]$ =

(C_j each for 1 point, p and g are 1.5 points for each)

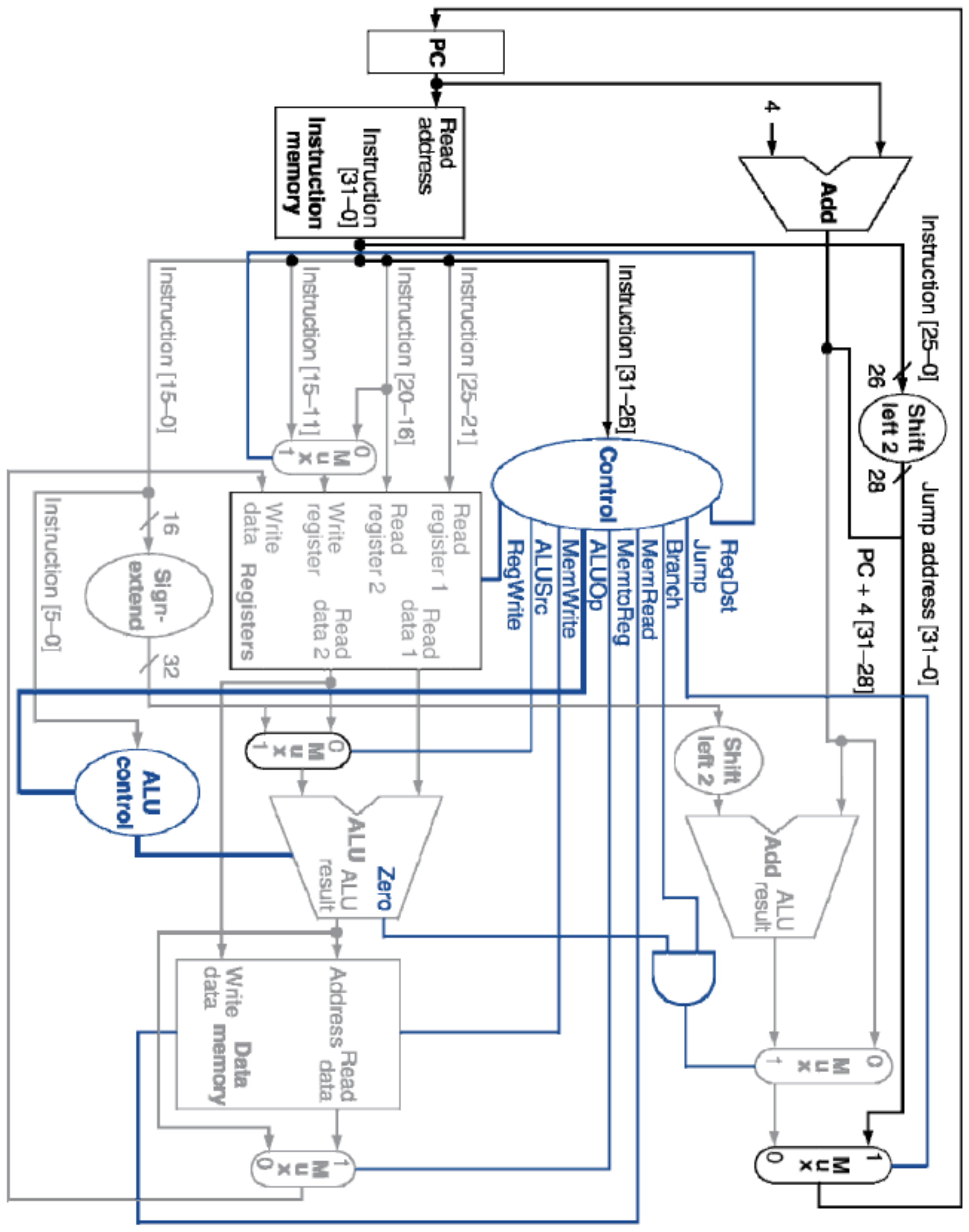
Problem 6 (12 points):

Consider the single-cycle processor implementation. Your task will be to augment this data path with a new instruction: the fmov instruction. The instruction will be an I-type instruction, and will have the following effect:

```
if (M[R[rs]]==SE(I))
        R[rt]=R[$t0]
```

Note that the fmov always uses register $t0 as the source value that is put into R[rt] - it is an implicit operand.

Implement your solution on the following two pages. All other instructions must still work correctly after your modifications. You should not add any new ALUs, register file ports, or ports to memory.


PC
4
Add
Read address
Instruction [31–0]
Instruction memory
Instruction [25–0]
26
Shift left 2
28
Jump address [31–0]
PC + 4 [31–28]
Instruction [31–26]
Control
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
Read register 1
Read register 2
Write register
Write data
Registers
Read data 1
Read data 2
16
Sign-extend
32
Shift left 2
Add
ALU result
ALU
Zero
ALU result
ALU control
Address
Read data
Write data
Data memory
M u x
0
1

Main Controller

Input or Output	Signal Name	R-format	lw	sw	Beq
Inputs	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Outputs	RegDst	1	0	X	X
	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

ALU Controller

opcode	**ALUOp**	**instruction**	**function**	**ALU Action**	**ALUCtrl**
lw	00	load word	XXXXXX	add	010
sw	00	store word	XXXXXX	add	010
beq	01	branch equal	XXXXXX	subtract	110
R-type	10	add	100000	add	010
R-type	10	subtract	100010	subtract	110
R-type	10	AND	100100	AND	000
R-type	10	OR	100101	OR	001
R-type	10	SLT	101010	SLT	111