CS 251, Fall 2016, Assignment 3.0.1

3% of course mark

Due Monday, October 31, 4:30 PM

1. (4 points)

   Consider the single-cycle computer shown on page 3 of this assignment. Suppose the circuit elements take the following times: Instr mem: 100ps, Register read: 30ps, ALU and all adders: 50ps, Register write: 30ps, Data memory: 100ps. Assume that PC and MUXes don't take any time.

   Compute the **minimum** cycle time for each instruction type below: 6

   R-format: 210 ps.

   LW: 310 ps.

   SW: 210 ps.

   Branch: 180 ps.

2. (8 points)

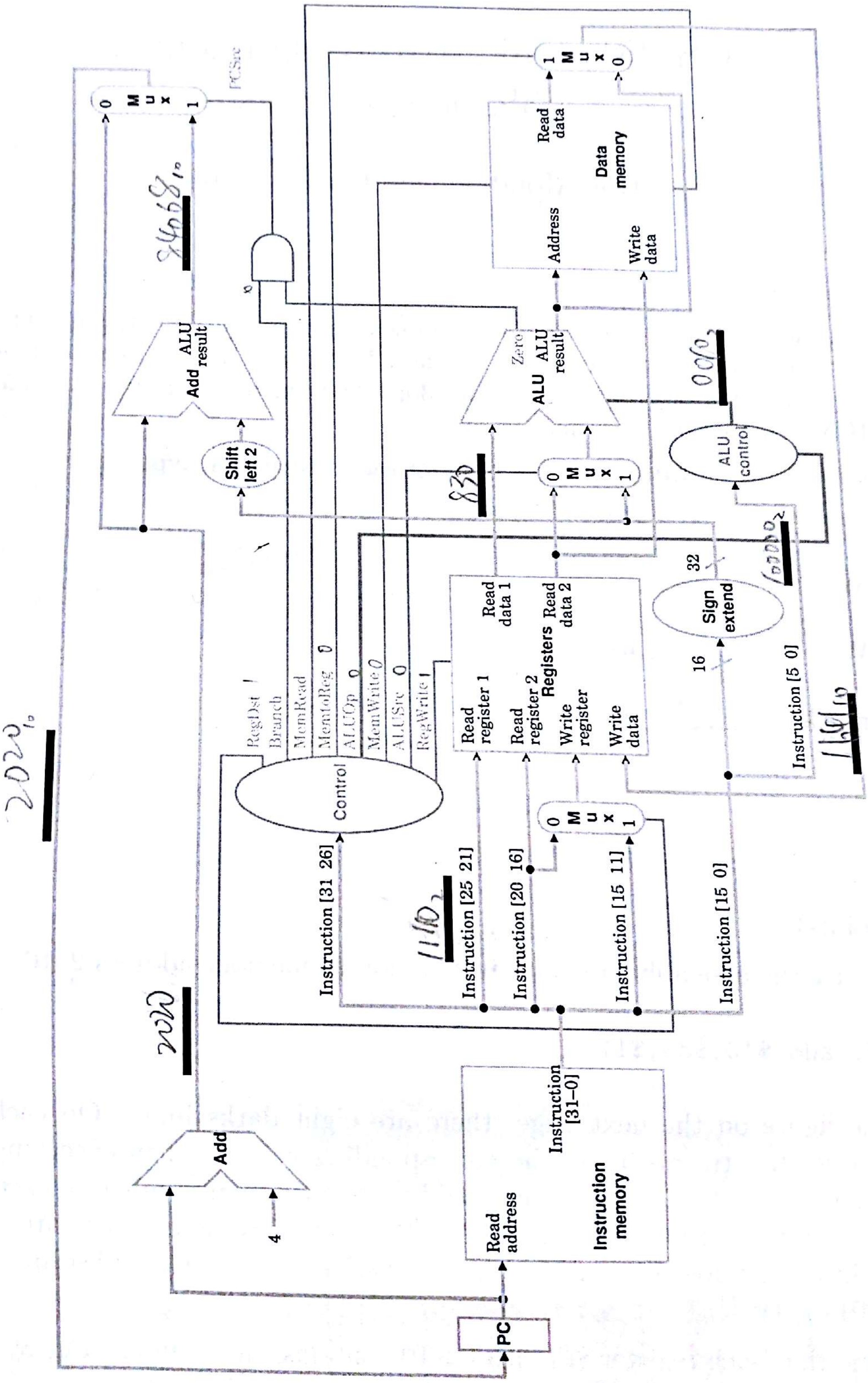
   Consider the assembly language instruction at memory address 2016:

```
2016: add $10,$30,$11
```

   In the figure on the next page, there are eight darks lines. On each line, write in the value that travels along the corresponding wire(s) when executing this assembly language instruction. Note: you should write a decimal number on each dark line, and **not** an expression involving things like 'PC', etc. (Some numbers are more natural to write in binary; for any binary numbers you use, you should subscript them with a '2' like 101_2.)

   Assume that each register $\$i$ (with $i > 0$) contains the decimal value $800_{10} + i$. Further assume that the `shamt` field of this R-format instruction is 0.

(Question 2, continued)



3. (13 pts) We want to modify the single-cycle computer to implement the R-format instruction **addjump**, or "add and jump". The form of this MIPS instruction is

```
addjump rd, rs, rt
```

This instruction adds the contents of registers rs and rt, stores the result in register rd, and jumps to the address rs+rt. The Funct bits for this instruction are 100001.

(a) (3 pts) The **addjump** instruction should use the ALU of the computer to perform the addition, which means that we have to modify the ALU Control unit. As a first step, we have reproduced to the truth table for the ALU control bits (Figure 4.13, page 261 of the course text) and added a new line for the **addjump** command:

ALUOp		Funct Field						Operation			
ALUop1	ALUop0	F5	F4	F3	F2	F1	F0	Op3	Op2	Op1	Op0
0	0	X	X	X	X	X	X	0	0	1	0
X	1	X	X	X	X	X	X	0	1	1	0
1	X	X	X	0	0	0	0	0	0	1	0
1	X	X	X	0	0	1	0	0	1	1	0
1	X	X	X	0	1	0	0	0	0	0	0
1	X	X	X	0	1	0	1	0	0	0	1
1	X	X	X	1	0	1	0	0	1	1	1
1	x	x	x	0	0	0	1	0	0	1	0

(We have broken the last column of the table from the book into four columns.) From inspection, we see that Op3=0. Given sum of product expressions for Op2, Op1, and Op0 in terms of ALUop1, ALUop0, F3, F2, F1, F0. Do not simply these expressions.

Op3 = 0

Op2 = $ALUop0 + ALUop1\,\overline{F_3}\,\overline{F_2}\,F_1\,\overline{F_0} + ALUop1\,F_3\,\overline{F_2}\,F_1\,\overline{F_0}$

Op1 = $\overline{ALUop1}\;\overline{ALUop0} + ALUop0 + ALUop1\,\overline{F_1}\,\overline{F_2}\,\overline{F_3}\,F_0 + ALUop1\,F_1\,\overline{F_2}\,\overline{F_3}\,F_0$
$+ ALUop1\,F_1\,F_2\,F_3\,F_0$
$+ ALUop1\,F_0\,\overline{F_1}\,F_2\,\overline{F_3}$

Op0 = $ALUop1\,\overline{F_0}\,F_1\,\overline{F_2}\,F_3 + ALUop1\,F_0\,\overline{F_1}\,F_2\,\overline{F_3}$

(b) (6 pts) Modify the single-cycle computer shown on page 6 to implement the R-format instruction `addjump`.

Summarize your changes on the next page.

(c) (4 pts) In the table below, give the settings of the control bits to implement the new `addjump` MIPS instruction. Use Don't Cares where appropriate. If you need an extra control line to implement this instruction or if you need to increase the number of bits in a control line, add additional columns to the table for the new control line, split a column to increase the number of bits in a control line, and in either case include a note below explaining the effect of the new/increased control line(s) on the datapath and what its setting should be for other instructions. Make sure you do not break any other instructions.

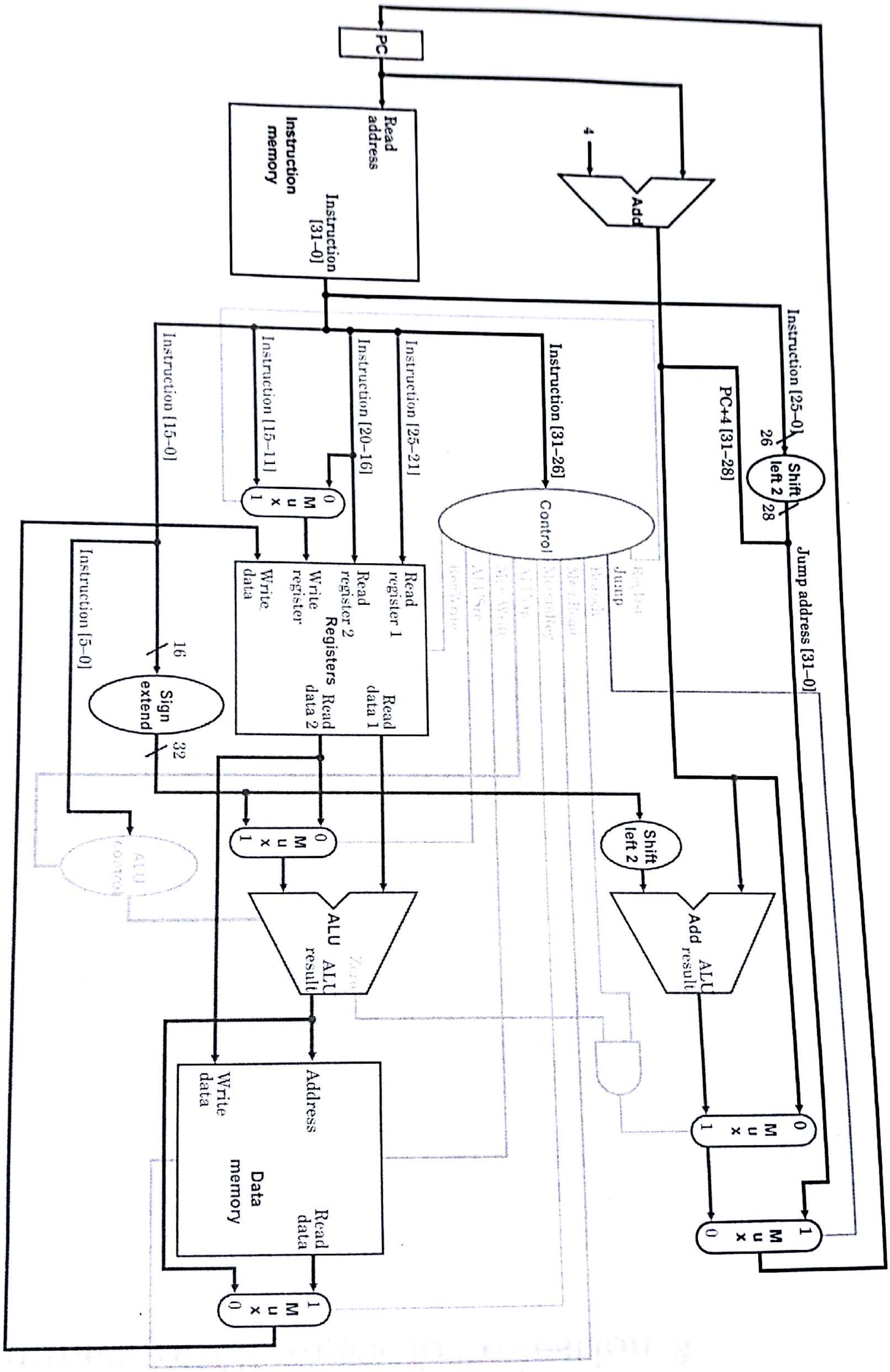
You will find tables of the effects of some of the control signals below. You should be able to determine the effects of the remaining signals from the datapath diagram.

Summarize your changes on the next page.

Type	Reg Dst	ALU Src	Mem ToReg	Reg Write	Mem Read	Mem Write	Branch	ALU op1	ALU op0	Jump
`addjump`	1	0	0	1	0	0	0	0	0	1

Signal	Signal=0	Signal=1
RegWrite	no effect	register written
MemRead	no effect	memory read
MemWrite	no effect	memory written

ALUOp	Operation
00	Add
01	Subtract
10	Funct Bits
11	—


PC
Read address
Instruction [31–0]
Instruction memory
4
Add
Instruction [25–0]
26
Shift left 2
28
Jump address [31–0]
PC+4 [31–28]
Instruction [31–26]
Control
Jump
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
M u x
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
ALU result
Address
Write data
Data memory
Read data