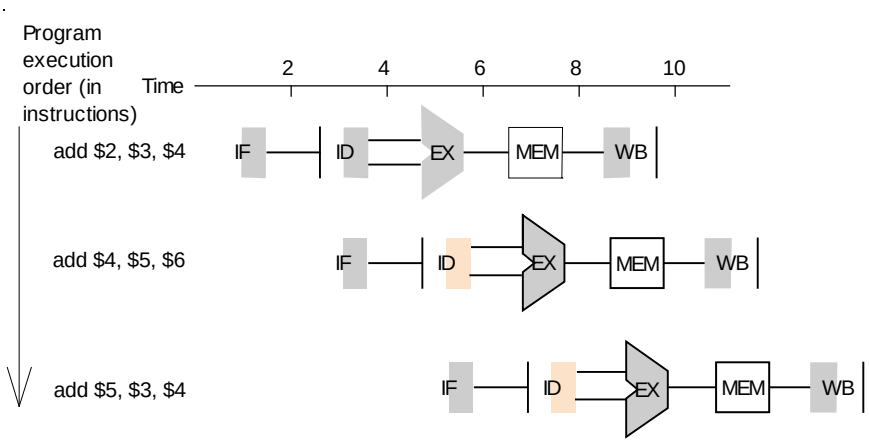
EC 362 Problem Set #4

1) Suppose we wish to add the new instruction `wai` (where am I) which puts the `wai` instruction's location (the value in the PC when the instruction was fetched) into a register specified by the rt field of the machine language instruction. Add any necessary datapaths and control signals to the multi-cycle machine in the figure below to implement this instruction. Also show any modifications / additions to the finite state machine diagram shown below.

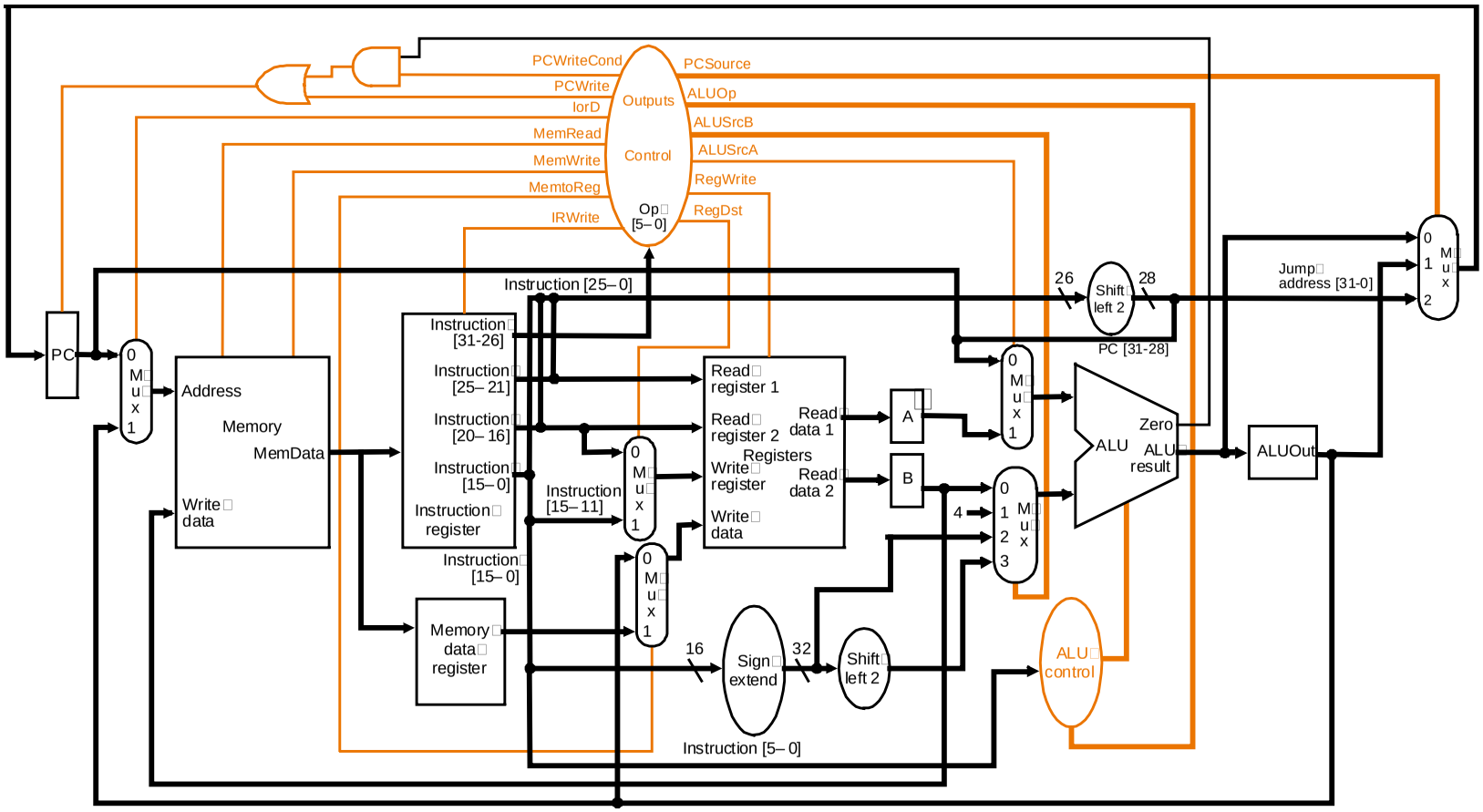
2) Suppose we wish to add the new instruction `jm` (jump memory). Its instruction format is similar to `lw`, except that the rt field is not used because the data loaded from memory is put into the PC instead of a target register (ex: jm 104($t0) would jump to the instruction whose address is found in the memory location 104+$t0). Add any necessary datapaths and control signals to the multi-cycle machine in the figure below to implement this instruction. Also show any modifications / additions to the finite state machine diagram shown below.

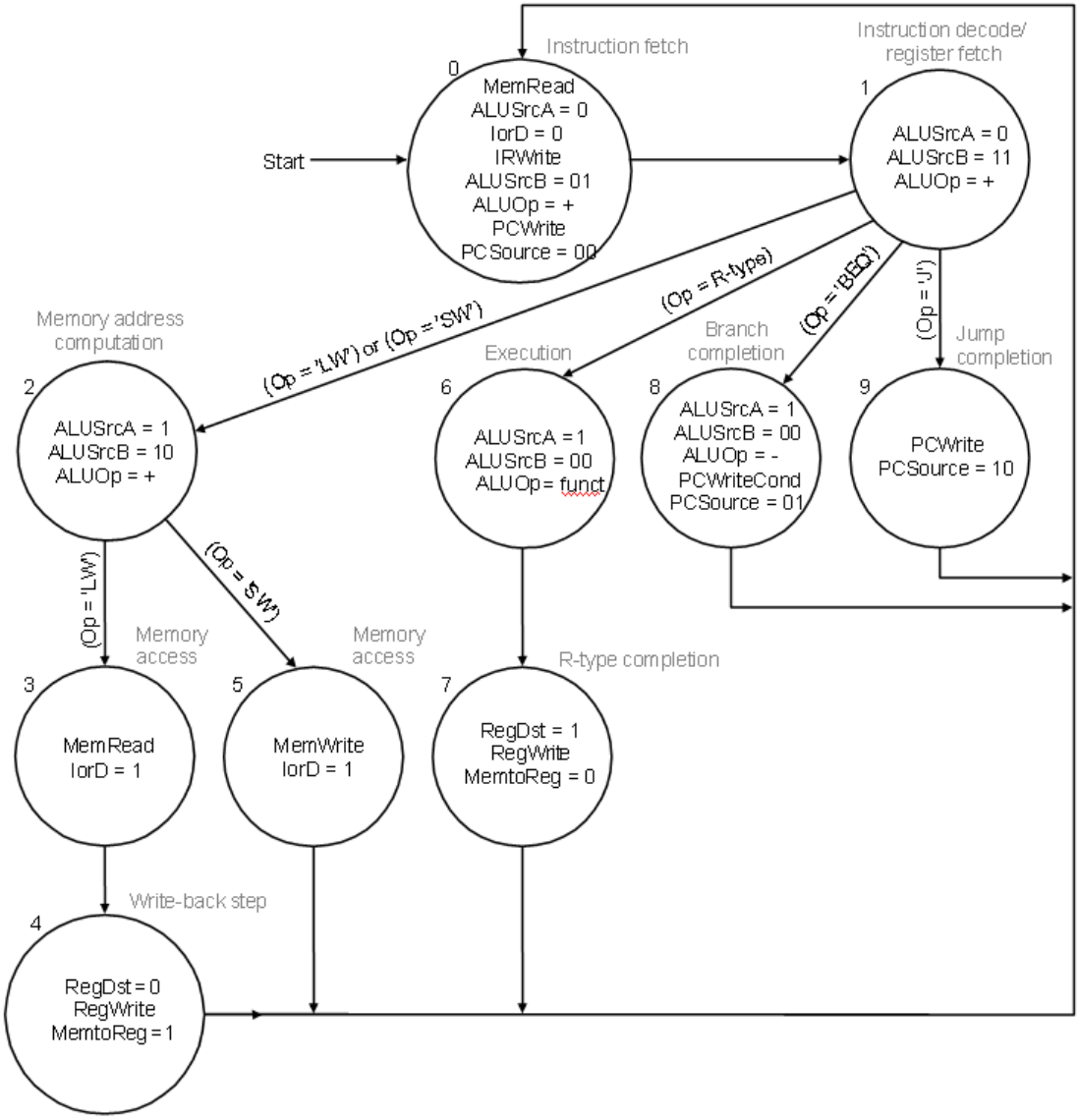
3) Using the following drawing, show the forwarding paths needed to execute the three instructions shown:



4) Modify the following code to make use a delayed branch slot, while retaining the same semantic intent.

```
loop: sw    $2, 100($3)
      addi  $3, $3, 4
      bne   $3, $4, loop
```


PCWriteCond
PCWrite
IorD
MemRead
MemWrite
MemtoReg
IRWrite
Outputs
Control
Op [5–0]
PCSource
ALUOp
ALUSrcB
ALUSrcA
RegWrite
RegDst
PC
0 M u x 1
Address
Memory
MemData
Write data
Instruction [31-26]
Instruction [25–21]
Instruction [20–16]
Instruction [15–0]
Instruction register
Instruction [15–0]
Instruction [25–0]
Instruction [15–11]
Memory data register
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
A
B
4
16
Sign extend
32
Shift left 2
Instruction [5–0]
26
Shift left 2
28
PC [31-28]
ALU
Zero
ALU result
ALUOut
ALU control
Jump address [31-0]
0 1 2 3 M u x


Instruction fetch
0
MemRead
ALUSrcA = 0
IorD = 0
IRWrite
ALUSrcB = 01
ALUOp = +
PCWrite
PCSource = 00
Start
Instruction decode/
register fetch
1
ALUSrcA = 0
ALUSrcB = 11
ALUOp = +
(Op = 'LW') or (Op = 'SW')
(Op = R-type)
(Op = 'BEQ')
(Op = 'J')
Memory address
computation
2
ALUSrcA = 1
ALUSrcB = 10
ALUOp = +
Execution
6
ALUSrcA = 1
ALUSrcB = 00
ALUOp = funct
Branch
completion
8
ALUSrcA = 1
ALUSrcB = 00
ALUOp = -
PCWriteCond
PCSource = 01
Jump
completion
9
PCWrite
PCSource = 10
(Op = 'LW')
(Op = 'SW')
Memory
access
3
MemRead
IorD = 1
Memory
access
5
MemWrite
IorD = 1
R-type completion
7
RegDst = 1
RegWrite
MemtoReg = 0
Write-back step
4
RegDst = 0
RegWrite
MemtoReg = 1