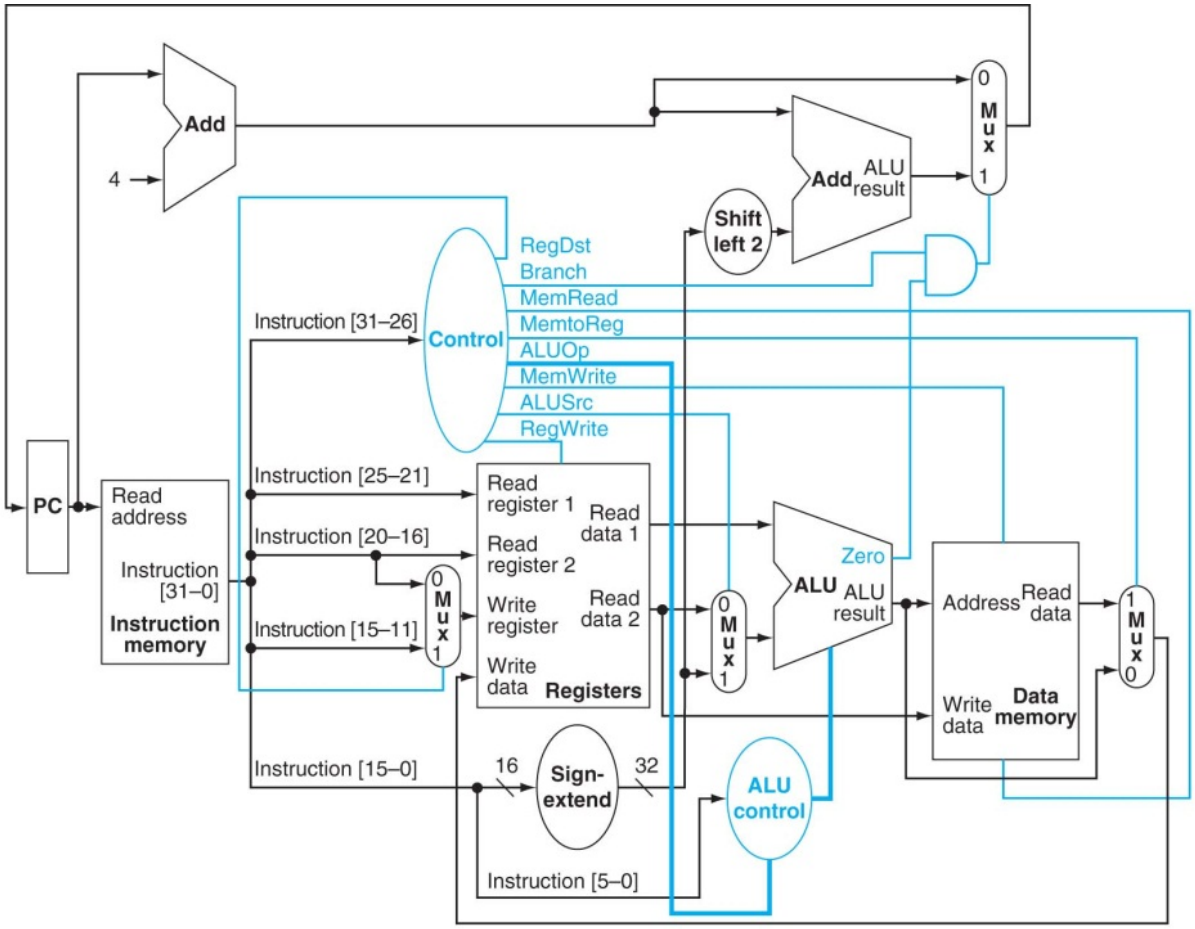
CSE 320 Midterm Practice - Single Cycle Datapath

In class we covered the MIPS single-cycle implementation which handled only a subset of the MIPS instructions: R-type (add, sub, and, or, slt), memory references (lw, sw), conditional branch (beq) and jump (j). In this problem we will add to the implementation functionality for additional MIPS instructions. Use this datapath and control table to specify your changes, add additional rows/columns to the control table as necessary. Be sure to mark don't cares in the control table whenever possible.

Problem Guidelines:

- When adding new instructions, don't break the operation of the standard ones.
- Avoid adding ALUs, adders, Reg Files, or memories to the datapath
- You can add MUXes, logic gates, etc. but try to do minimally. (these cost in terms of area, cycle time, etc)



Instruction	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUOp0	Jump
R-format	1	0	0	1	0	0	0	1	0	0
`lw`	0	1	1	1	1	0	0	0	0	0
`sw`	X	1	X	0	0	1	0	0	0	0
`beq`	X	0	X	0	0	0	1	0	1	0
`J`	X	X	X	0	X	0	0	X	X	1

a. Mark/Highlight the datapath used for each of the following instructions types: `lw, sub, beq, j`

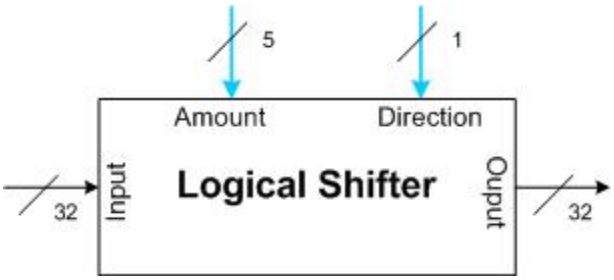
b. Modify the datapath and the control table to implement a new type of instruction 'lwx' (load word - indexed), which in assembly is `lwx rd, rs(rt)`

```
Reg[rd] <- Mem[ Reg[rs] + Reg[rt] ];
# register rd is loaded with the memory location specified by the address rs+rt
```

b. Modify the datapath and the control table to implement logical shifting instructions, 'srl' and 'sll'. A shifter may be added to the datapath as specified below.

```
Reg[rd] <- Reg[rt] >> shamt;
# register rt is shifted to the right by the shamt. The upper bits are loaded with 0.
Reg[rd] <- Reg[rt] << shamt;
# register rt is shifted to the left by the shamt. the lower bits are loaded with 0.
```

The shifter amount control specifies the number of bits to shift. The direction control is 0 to shift to the right and 1 to shift to the left. The value to be shifted is placed on the input and after a delay the shifted value is available on the output.



c. Modify the datapath and the control table to implement a new type of instruction 'sw+', "storeword-and-autoincrement". This function is useful if we have to store many elements in an array and have to run through it quickly. It stores a word just like 'sw' does, but it also increments the address register to the next word in the same single instruction. For example, `sw+ $8, 0($9)` will store the current value of register $8 into memory location pointed to by register $9 (plus the offset 0). It will then increment the address in register $9 by 4 to point to the next word in the memory. Thus, the 'sw+' instruction is equivalent to the two MIPS instructions:

```
sw $8, 0($9)
addi $9, $9, 4
```

Note that it will always auto increment by 4, and the offset does not have to be 0, we could have said `sw+ $8, 20($9)`.

In essence, the datapath should be performing 'sw' and 'addi' in the parallel in the same clock cycle. Assume that 'sw+' has the same instruction format as 'sw' but a different opcode.

Single-cycle datapath timing

a. Calculate the delay in the modified datapath when performing instructions b-d in Problem 1. Assume the following delays:

- Memory: 200ps
- Register Files Access: 50ps
- ALU and adders: 100ps
- Shifter: 30ps

b. Calculate the longest path for all instructions in the modified datapath considering the addition of all instructions. IE for the full modified datapath what is the minimal cycle time?