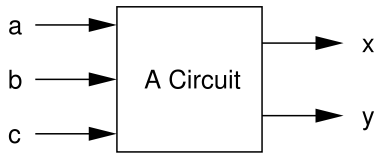
1 Digital Logic Design

1. Suppose you were asked to build a circuit that consists of three inputs a, b, and c, and two outputs x and y as shown below (on left). The truth table of this circuit is shown below (on right):



Inputs			Outputs	
a	b	c	x	y
0	0	0	0	1
0	0	1	0	0
0	1	0	1	1
0	1	1	1	0
1	0	0	0	0
1	0	1	0	1
1	1	0	1	0
1	1	1	1	1

What is the **boolean expressions** representing outputs x and y **using Karnaugh map** to simplify the circuit? You must draw the Karnaugh maps for both outputs x and y. Also draw the final circuit using only AND, OR, and NOT gates.

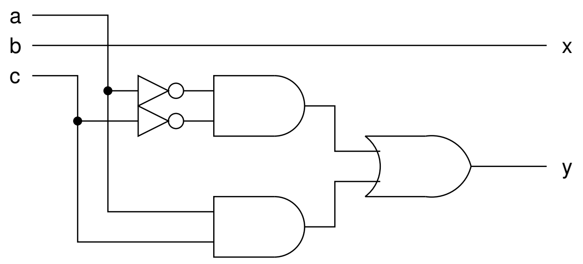
Solution:

c ab	0	1
00	0	0
01	1	1
11	1	1
10	0	0

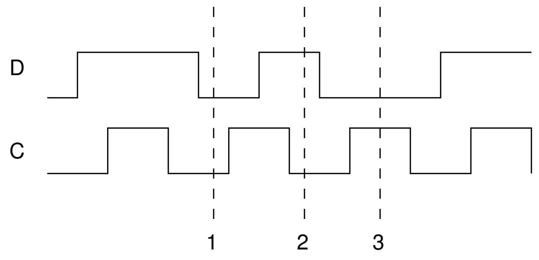
Output $x = b$

c ab	0	1
00	1	0
01	1	0
11	0	1
10	0	1

Output $y = a'c' + ac$



2. Consider a D flip flop as we discussed in class. The graph below shows inputs D and C of a D flip-flop.



What are values of output Q at labels 1, 2, and 3?

	1	2	3
Value	1	1	0

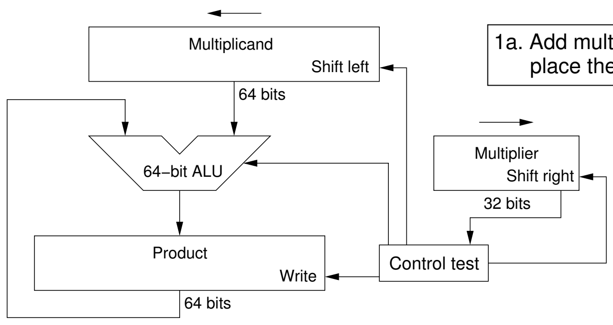
2 Arithmetic in Computer

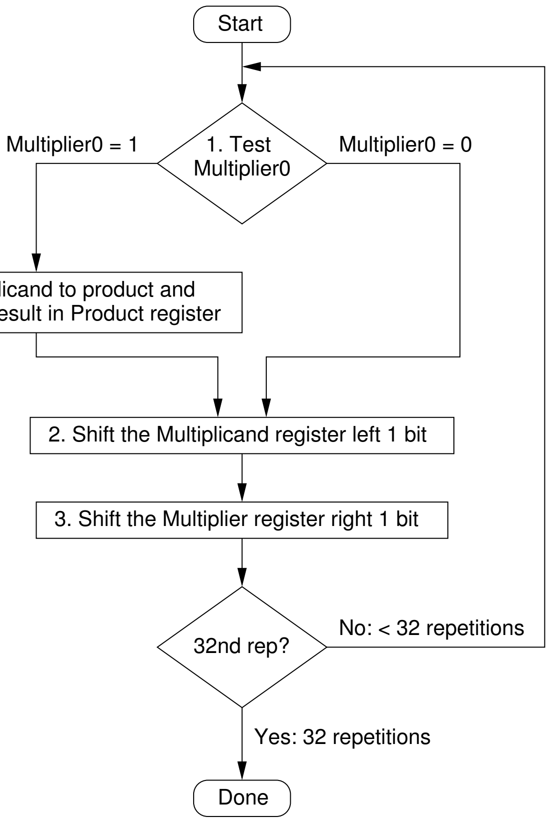
1. Using paper and pencil method, show how to multiply 1010_2 (-6_{10}) by 0110_2 (6_{10}). Note that those number are in 4-bit representation **signed** numbers in two's complement.

Solution: To support signed multiplication using paper and pencil method, we need to extend 4-bit numbers into 8-bit numbers before multiplication.

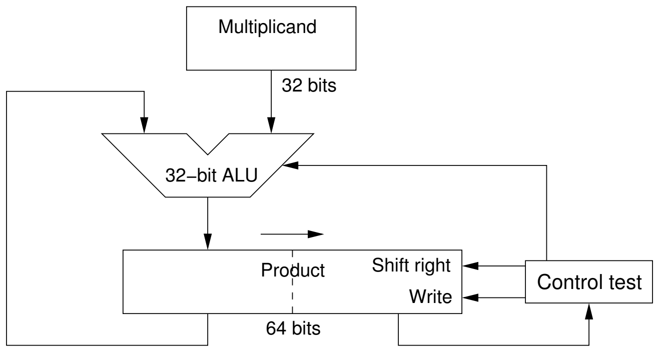
```
11111010
00000110 x
--------
00000000
1111010
111010
00000
0000     +
000
00
0
--------
11011100
========
```

2. Consider a circuit diagram and its flowchart below:



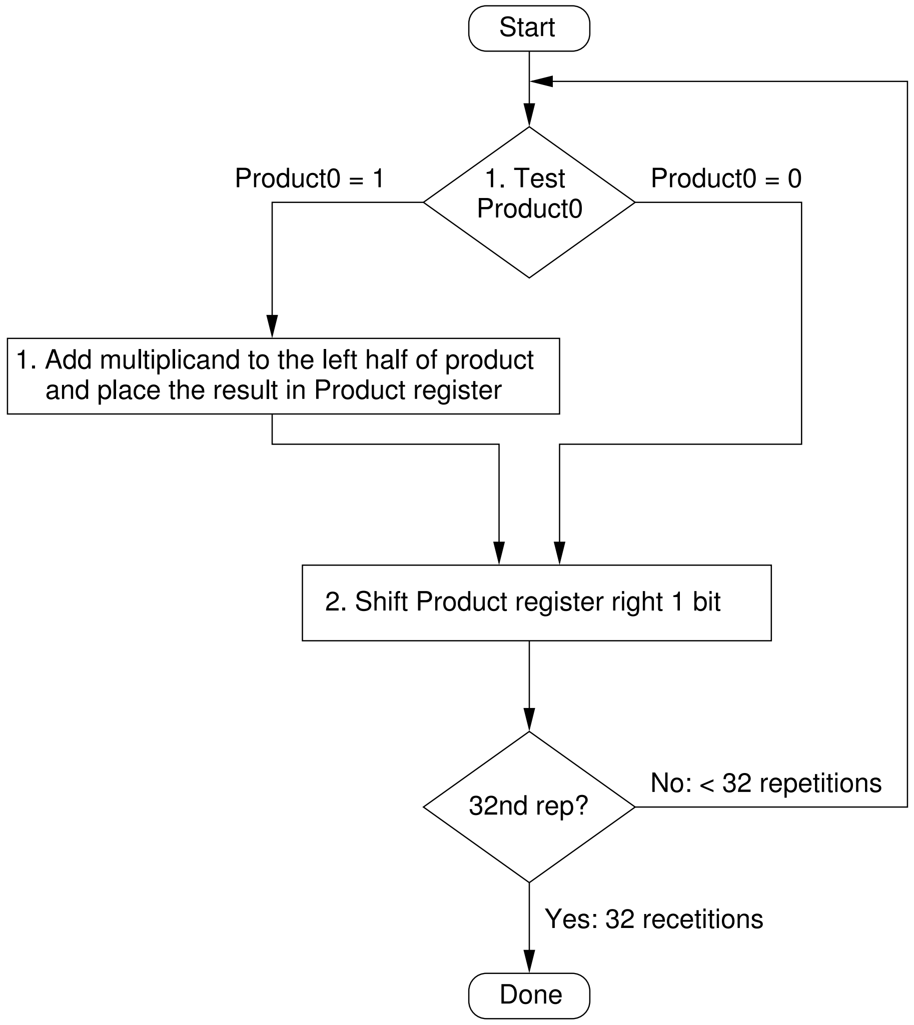

Start
1. Test Multiplier0
Multiplier0 = 1
Multiplier0 = 0
1a. Add multiplicand to product and place the result in Product register
2. Shift the Multiplicand register left 1 bit
3. Shift the Multiplier register right 1 bit
32nd rep?
No: < 32 repetitions
Yes: 32 repetitions
Done

The figures above show a multiplication hardware and its multiplication algorithm for two 32-bit unsigned number discussed in class. The figure below shows a refined version of the multiplication hardware above. For this refined hardware version, before the multiplication process starts, the multiplier will be located at the lower 32-bit of product register, upper 32-bit of product register will be 0s, and the control test only see the least significant bit of the product register called `Product0`.



Draw the flowchart of the multiplication algorithm for the **refined version** shown above.

Solution:


Start
1. Test Product0
Product0 = 1
Product0 = 0
1. Add multiplicand to the left half of product and place the result in Product register
2. Shift Product register right 1 bit
32nd rep?
No: < 32 repetitions
Yes: 32 recetitions
Done

3 Floating-Point Representation

1. What is the value of 14.875_{10} in binary with binary point?

   Solution: 14_{10} is 1110_2. $0.875 \times 2 = 1.75$, $0.75 \times 2 = 1.5$, and $0.5 \times 2 = 1.0$. Thus 14.875_{10} is 1110.111_2.

2. What is the value of 1100.1011_2 in decimal (not in scientific notation)?

   Solution:

   $$(1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3}) + (1 \times 2^{-4})$$

   which is equal to $8 + 4 + 0.5 + 0.125 + 0.0625 = 12.6875$

3. How to represent the binary number with binary point 1100.1011010010110_2 in IEEE 754 format (32-bit single precision)?

   Solution: $1100.1011010010110 = 1.1001011010010110 \times 2^3$. Thus $3 = e - 127$ or $e = 130$ which is 10000010_2. The above number in IEEE 754 format is

   $$0\ 10000010\ 1001011010010110 \ldots 0$$

4. A 32-bit binary (single precision) in IEEE 754 format is shown below:

   $$1\ 10000000\ 11100000000000000000000$$

   What is the above floating-point number in decimal?

   Solution: From the above number e is 128. Thus the number is

   $$(-1)^1 \times (1 + 0.111) \times 2^{128-127} = -1.111 \times 2^1 = -11.11$$

   Thus, the number is -3.75

5. Consider the following two floating-point number a and b in IEEE 754 format (single precision):

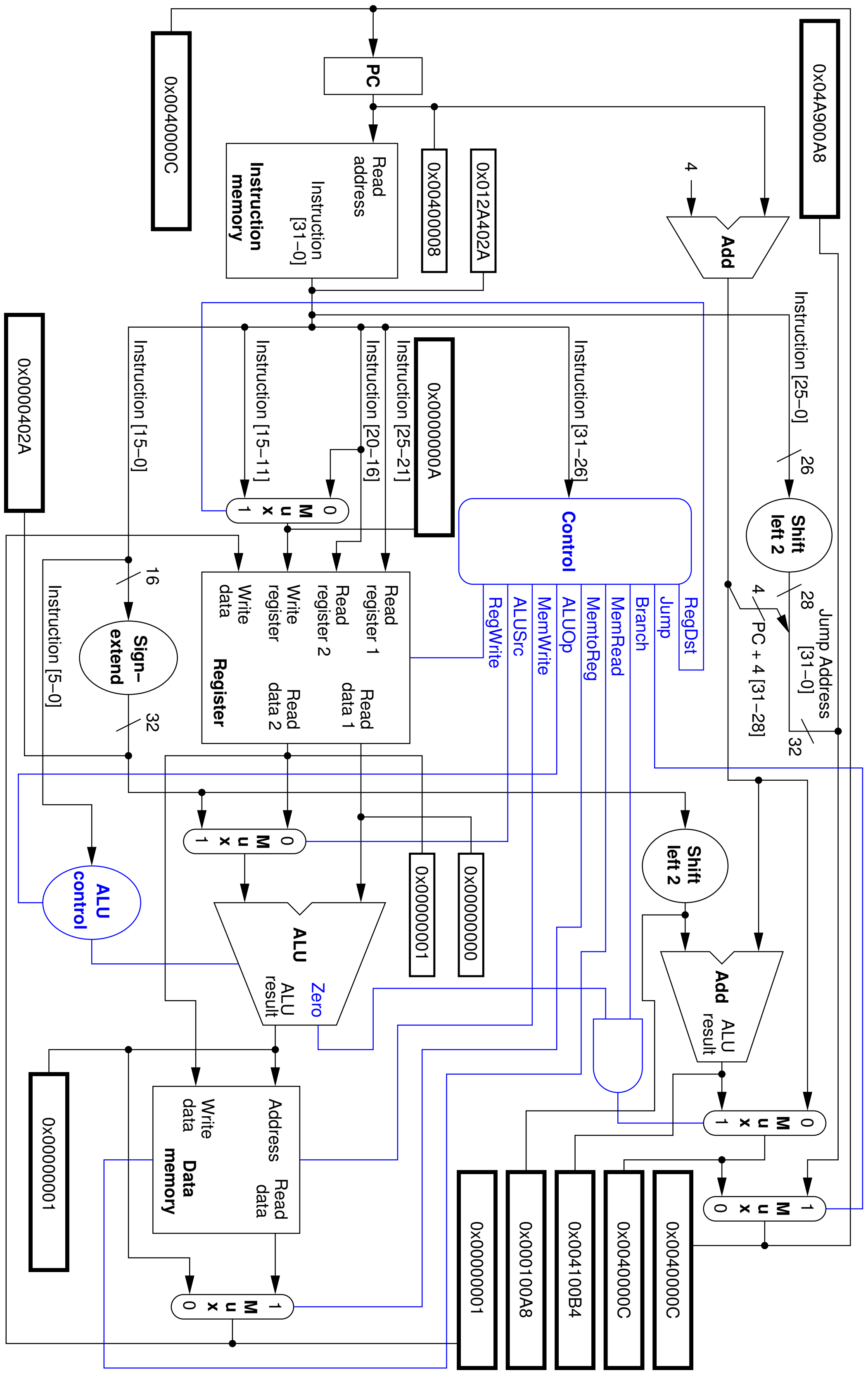
   $$a = 0\ 10101010\ 11001100110011001100110$$

   $$b = 0\ 10101011\ 00110011001100110011001$$

   Which number is larger and why?

   Solution: b is larger since the exponent is larger.

MIPS Datapath



4 Processor

According to the processor diagram shown in page 6, answer the following questions:

1. What are values of all control signals (0 or 1) except `ALUOp` when the instruction `and` is being executed?

`RegDst`	`Jump`	`Branch`	`MemRead`	`MemtoReg`	`MemWrite`	`ALUSrc`	`RegWrite`
1	0	0	0	0	0	0	1

2. What are values of the following control signals (0 or 1) when the instruction `beq` is being executed?

`RegDst`	`MemRead`	`MemtoReg`	`MemWrite`	`ALUSrc`	`RegWrite`	`Ainvert`	`Bnegate`
0/1	0	0/1	0	0	0	0	1

3. When we design the CPU shown in page 6, we did not consider the instruction `slti`. Luckily, our processor actually supports `slti` by simply modify the control unit. Note that `slti` is an I-type. What are values of the following control signals (0 or 1) if we want our CPU to execute `slti` correctly?

`RegDst`	`MemRead`	`MemtoReg`	`MemWrite`	`ALUSrc`	`RegWrite`	`Ainvert`	`Bnegate`
0	0	0	0	1	1	0	1

   Solution: `add`, `sub`, `and`, `or`, `slt`, `beq`

4. In the processor diagram shown in page 6, there are 10 empty boxes around the CPU. You can think of these boxes as output components in Logisim to monitor values of various places in the CPU. Suppose at an instance of time, the ouput of the program counter (PC) is 0x00400008, the instruction stored in the instruction memory at the location 0x00400008 is `slt $t0, $t1, $t2` (0x012A402A), the value stored in the register `$t1` is zero, and the value stored in the register `$t2` is 1. Fill in all empty boxes on page 6 using **hexadecimal** values. If the value is only 5 bits, pad the upper 27 bits by 0s. Make sure you see all 10 boxes.

5 MIPS Processor Design (10 Points)

Recall that the CPU design (shown in page 6) does not support `jal` (jump and link) instruction. According to the definition of the instruction `jal`, it stores the return address (`PC + 4`) in the register `$ra` (register number 31_{10}) and makes `PC = {PC + 4[31:28], address, 2'b0}` where `address` is the address field in J-type. **Explain in words, what do we need to modify our CPU to support the `jal` instruction**. For example,

- Do we need any extra component(s)? If yes, explain what they are and why?
- Do we need any extra control line(s)? If yes, explain what they are and why?

Again, simply explain your idea in words. You do not have to draw the circuit diagram. **HINT**: the definition of the instruction `j` (jump) is `PC = {PC + 4[31:28], address, 2'b0}`.

Solution: Since we need to be able to write PC + 4 to the register `$ra`, we need a multiplexer right before the "Write data" of the register file. For this multiplexer, when the select line is 1, it will select PC + 4 to the "Write data". Otherwise, it will select the output that come from the multiplexer controlled by "MemtoReg" signal. This multiplexer should be controlled by a new control line named jal. This control line will be 1 when the instruction is `jal`. Otherwise, this control line should be 0.

We also need another multiplexer right before "Write register". For this multiplexer, when its select line is 1, it should select a constant 11111_2. Otherwise, it will select the output that come from the multiplexer controlled by "RegDst" signal. We also need another control line to control this multiplexer. May be called "raDst". This control line should be 1 when the instruction is `jal`. Otherwise, it should be 0.

For the existing control lines, they should be the same as the instruction `j` except "RegWrite". The "RegWrite" control signal should be 1.