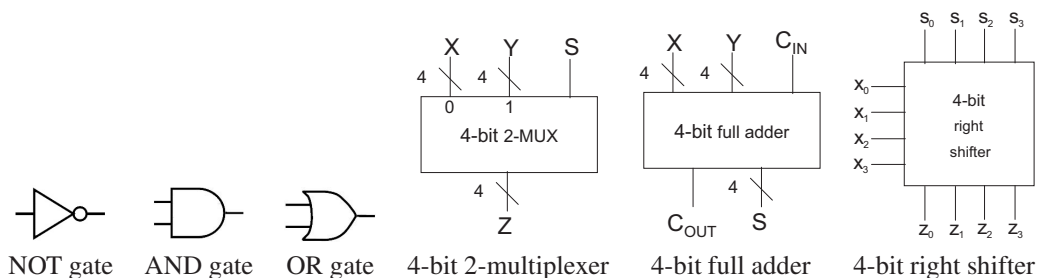


Midterm Exam

November 12, 2007

- Use * to represent the irrelevant values and the values not specified in the question.
- Please use the following notations for logic gates. Feel free to change the orientations of the gates and the positions of inputs and outputs. You are free to use other circuits. However, if they are not introduced in the class, you have to implement them before using them.



hlt rd, addr	add rd, rs, rt st rd, addr	sub rd, rs, rt ldi rd, rt	and rd, rs, rt sti rd, rt	xor rd, rs, rt bz rd, addr	shl rd, rs, rt bp rd, addr	shr rd, rs, rt jr rd	lda rd, addr jl rd, addr
------------------------	---	--	--	---	---	---------------------------------------	---

- (6%) What are the 8-bit 2's complement representations of the following decimal numbers? Please give both their binary and hexadecimal representations.
 - 10
 - 97
- (8%) Implement a circuit to realize the 3-bit *mod3* function. The circuit has a 3-bit input $X = X_2X_1X_0$ and a 2-bit output $Y = Y_1Y_0$ satisfying $Y = X \% 3$. For example, if X is 101, then Y is 10 because $5 \% 3 = 2$. Write down the truth table and its corresponding logic expressions for Y_1 and Y_0 .
- (18%) In the class, we have implemented the 4-bit right-shifter. Note that, to be consistent with the lecture, let Z_0 and X_0 represent the MSBs. (a) Modify the right-shifter so that it implements sign extension. Write down the corresponding logic expressions for Z_0, Z_1, Z_2 and Z_3 . (b) Please implement a 4-bit left-shifter whose diagram is shown as Figure 1(a). It has two 4-bit inputs, $X = X_0X_1X_2X_3$ and $S = S_0S_1S_2S_3$, and a 4-bit output Z . Note that only one of S_i will be on at a time and $Z = X \ll i$ when $s_i = 1$. Write down the logic expression for Z_i . (c) Use your 4-bit left-shifter and any circuits defined in the class to implement a 4-bit left/right-shifter (Figure 1(b)). The circuit has a 4-bit input X , a 1-bit input d and a 2-bit input S , where d specifies the shift direction. When $d = 1$, the shift is to the left. Please draw the circuit. (No need to optimize.)
- (8%) Please explain why we need copy input 2 in TOY ALU. Hint: give the instructions which need it.
- (8%) Why did we change the definition of **jr** from $PC \leftarrow R[d]$ to $PC \leftarrow R[t]$?
- (10%) Draw the required datapath on the accompanying answer sheet (Figures 4 and 5) for (a) instruction fetch (b) the set of ALU instructions (opcode 1-6) and **lda**. Please only draw the necessary part but not the whole TOY datapath. Add multiplexers if necessary.
- (10%) According to the TOY architecture (Figure 2), write down the logic expression for MUX_{MEM} and MUX_{REGW} in terms of inputs to control (clock, execute, fetch, cond_pos, cond_zero and 16 instructions (hlt, add, sub, ...)).

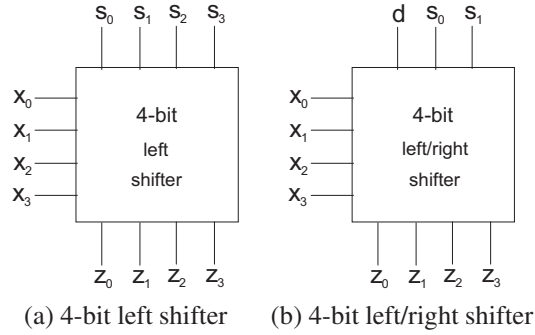


Figure 1: Shifter.

8. (12%) Refer to the TOY architecture (Figure 2), please specify the operations of MUX_{PC} , MUX_{MEM} , MUX_{REGR} , MUX_{ALU} , MUX_{REGW} , $WRITE_{REG}$, $WRITE_{MEM}$ and ALU_{OP} during the execution stage for the instructions "xor", "store" and "load indirect". For example, the answer for "jump and link" would be $MUX_{PC} = 0$, $MUX_{MEM} = *$, $MUX_{REGR} = *$, $MUX_{ALU} = 1$, $MUX_{REGW} = 01$, $WRITE_{REG} = 1$, $WRITE_{MEM} = 0$, $ALU_{OP} = *$.
9. (12%) Please trace the content of RA, RB and RC along time for the following TOY assembly program. And, what is the final output to stdout?

```

        lda RA, 12
        lda RB, 8
repeat  sub RC, RA, RB
        bz RC, exit
        bp RC, positive
        sub RB, RB, RA
        bz R0, repeat
positive sub RA, RA, RB
        bz R0, repeat
exit    st RA, 0xFF

```

10. (8%) Assume that we have three TOY object codes A.obj, B.obj and C.obj. Their sizes are 18, 32 and 23 words for A, B and C respectively. Assume that program B.obj exports the symbol `proc_b` and its address is 0x10 in B.obj. Similarly, program C.obj exports the symbol `var_c` whose address is 0x0A in C.obj. Assume that the source code for A.obj includes the following two instructions:

```

lda RA, var_c
jl  RF, proc_b

```

After linking using "toylink A.obj B.obj C.obj > A.toy", what are the corresponding machine code for the above two statements in the resulted TOY executable code?

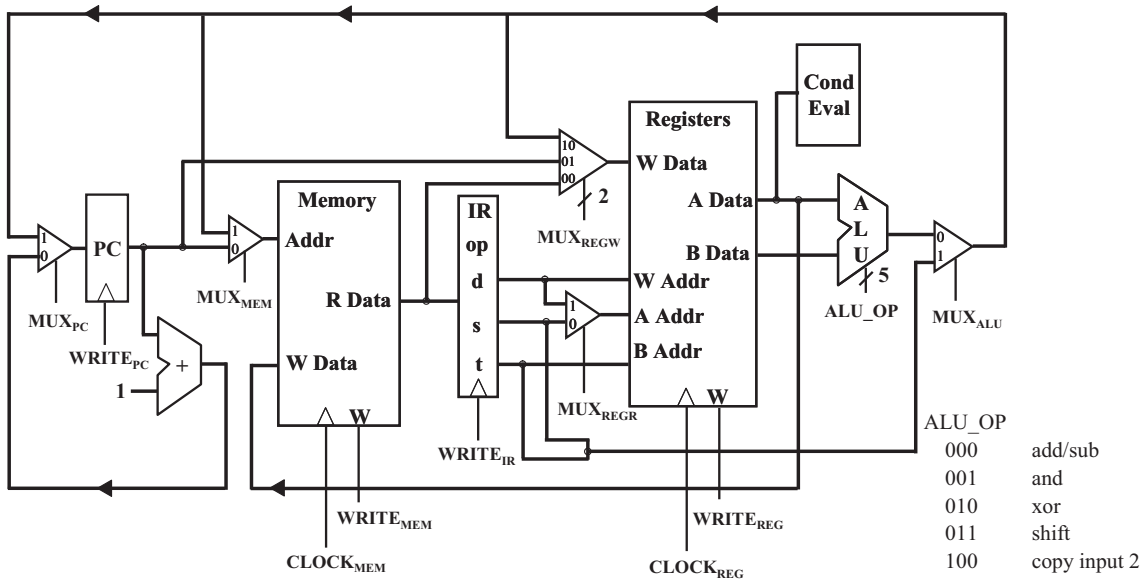


Figure 2: TOY architecture.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Format 1	opcode				dest d			source s			source t					
Format 2	opcode				dest d			addr								

#	Operation	Fmt	Pseudocode
0:	halt	1	exit(0)
1:	add	1	$R[d] \leftarrow R[s] + R[t]$
2:	subtract	1	$R[d] \leftarrow R[s] - R[t]$
3:	and	1	$R[d] \leftarrow R[s] \& R[t]$
4:	xor	1	$R[d] \leftarrow R[s] \wedge R[t]$
5:	shift left	1	$R[d] \leftarrow R[s] \ll R[t]$
6:	shift right	1	$R[d] \leftarrow R[s] \gg R[t]$
7:	load addr	2	$R[d] \leftarrow \text{addr}$
8:	load	2	$R[d] \leftarrow \text{mem}[\text{addr}]$
9:	store	2	$\text{mem}[\text{addr}] \leftarrow R[d]$
A:	load indirect	1	$R[d] \leftarrow \text{mem}[R[t]]$
B:	store indirect	1	$\text{mem}[R[t]] \leftarrow R[d]$
C:	branch zero	2	if ($R[d] == 0$) $\text{pc} \leftarrow \text{addr}$
D:	branch positive	2	if ($R[d] > 0$) $\text{pc} \leftarrow \text{addr}$
E:	jump register	1	$\text{pc} \leftarrow R[t]$
F:	jump and link	2	$R[d] \leftarrow \text{pc}; \text{pc} \leftarrow \text{addr}$

Register 0 always 0.
 Loads from mem[FF]
 from stdin.
 Stores to mem[FF] to
 stdout.

Figure 3: TOY reference card.

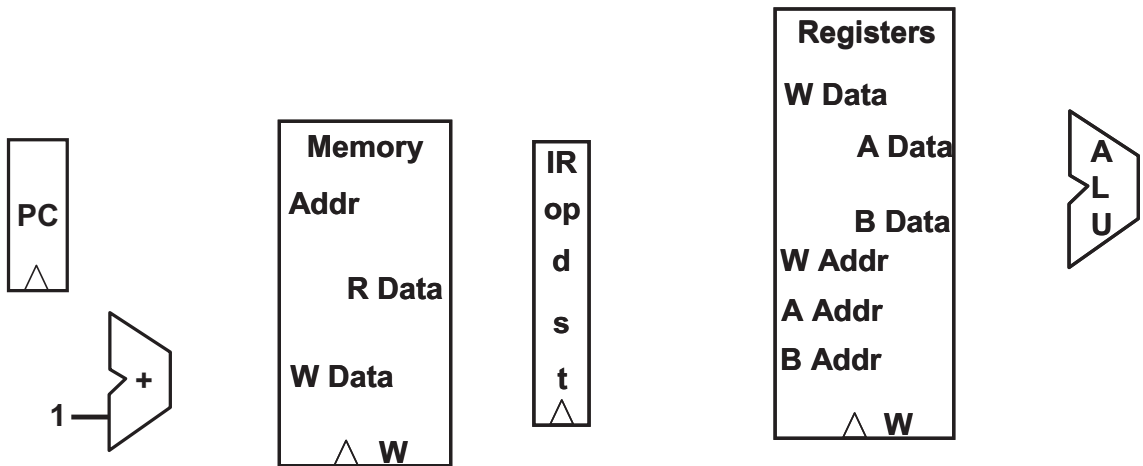


Figure 4: Datapath for question 6(a) (instruction fetch).

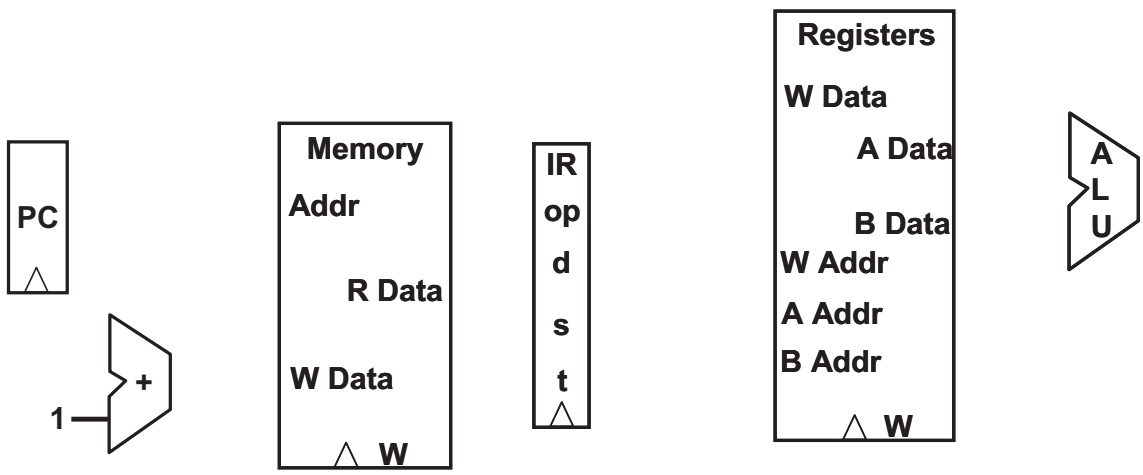


Figure 5: Datapath for question 6(b) (ALU instructions and ldr).