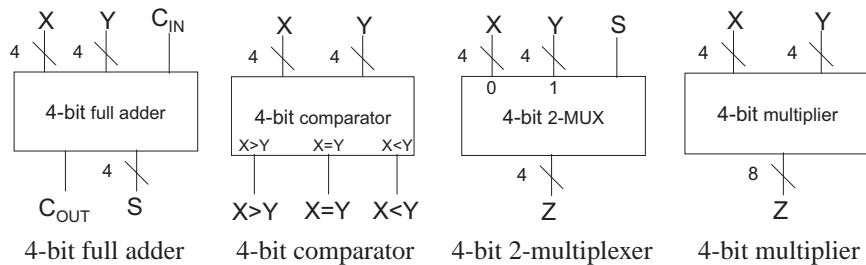
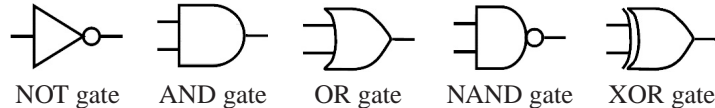


Midterm Exam

November 21, 2006

- 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.



opcode	mnemonic	syntax	opcode	mnemonic	syntax
0	hlt	hlt	8	ld	ld rd, addr
1	add	add rd, rs, rt	9	st	st rd, addr
2	sub	sub rd, rs, rt	A	ldi	ldi rd, rt
3	and	and rd, rs, rt	B	sti	sti rd, rt
4	xor	xor rd, rs, rt	C	bz	bz rd, addr
5	shl	shl rd, rs, rt	D	bp	bp rd, addr
6	shr	shr rd, rs, rt	E	jr	jr rd
7	lda	lda rd, addr	F	jl	jl rd, addr

opcode	rd	rs	rt	opcode	rd	addr	
--------	----	----	----	--------	----	------	--

Table 1: TOY instruction and assembly format.

- (10%) Convert the following into single-precision IEEE 754 numbers and display them in hexadecimal.
 - $-\frac{23}{64}$
 - $0.0101_2 \times 2^{-130}$
- (6%) Apply *DeMorgan's Theorem* to the expression, $\overline{A + \overline{B} + C + \overline{A}B}$, so that the NOT bars do not span more than a single variable (4%). Explain why this function always returns *false* (2%).
- (8%) Consider a 3-input Boolean function that returns *true* only when exactly two of its inputs are *true*. (a) Construct its truth table (4%). (b) Implement this function with logic gates AND, OR and NOT (4%).
- (6%) TOY ALU uses five control bits to specify its operation (Figure 1(a)). Since there are only seven distinct operations in total, it is possible to specify ALU operations using only three control bits. What change do you have to make to use only three bits to specify TOY ALU operations? What is its downside?

5. (12%) Refer to Figure 1(b) for TOY architecture. Please specify the operations of MUX_A , MUX_B , MUX_C , MUX_D , MUX_E , REG_W , MEM_W and ALU_{OP} during the execution stage for the instructions "load address", "store indirect" and "branch zero". Specify * for the irrelevant ones. For example, the answer for "jump and link" would be $MUX_A = 0$, $MUX_B = *$, $MUX_C = *$, $MUX_D = 1$, $MUX_E = 01$, $REG_W = 1$, $MEM_W = 0$, $ALU_{OP} = *$.

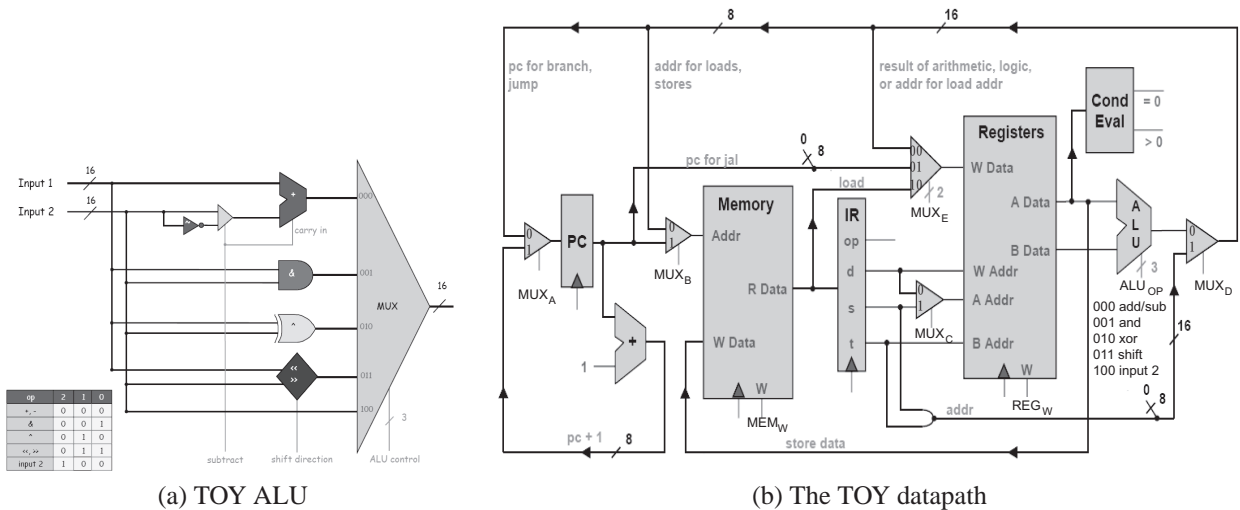


Figure 1: TOY architecture.

6. (18%) programming assignment?
7. (10%) Assume that the address of arrayA is 0. (a) what is the content of the memory? Specify it as a byte string and represent each byte in hexadecimal. What are the values of (b) size, (c) TYPE arrayB, (d) LENGTHOF arrayB, (e) SIZEOF arrayB?
- ```

arrayA BYTE 24, 0, -15
ALIGN 4
arrayB WORD 1234h, 5678h
 WORD 1357h, 2468h
size = ($ - arrayA)

```
8. (10%) Answer the following questions:
- Is C0A5h a valid hexadecimal constant?
  - After executing "movsx eax, -4", what is the content of EAX?
  - The instruction, inc [esi], won't assemble correctly. Why and suggest a way to fix it.
  - If AL contains +127 and you add 3 to AL, will the Carry and Overflow flags be set?
  - The CALL instruction pushed the offset of the CALL instruction on the stack.
9. (5%) What is the content of EAX after executing the following code?
- ```

MOV    EAX, 0
MOV    ECX, 5
L1:   ADD    EAX, 10
      MOV    ECX, 3
L2:   INC    EAX
      LOOP  L2
      LOOP  L1

```
10. (10%) Design a binary multiplier that multiplies two 4-bit numbers as shown in the multiplier block diagram at the beginning, where $X = X_3X_2X_1X_0$, $Y = Y_3Y_2Y_1Y_0$ and $Z = X \cdot Y = Z_7Z_6 \dots Z_0$, in which X_0 , Y_0 and Z_0 are LSBs. You may use the notation $X[n..m]$ to identify a portion of wires. For example, $X[3..1]$

means the set of wires, $X_3X_2X_1$. (Note that, you do not actually need to use shifters and 8-bit adders to implement the multiplier. If you do, two points may be deducted.)