University of California, Berkeley
College of Engineering
Computer Science Division — EECS

Spring 1998 J. Wawrzynek

CS152 Computer Architecture and Engineering
Midterm I

Your Name: ______________________________

ID Number: ______________________________

This is a *closed-book, closed-note* exam. No calculators. You have 3 hours. Each question is marked with its number of points (one point per expected minute of time).

Show your work. Write neatly and be well organized.
Good luck!

problem	maximum	score
1	20pts	
2	20pts	
3	15pts	
4	15pts	
5	30pts	
total	100pts	

1. [20 points] Short answers. Please provide *short* answers to the following questions. Correct answers are worth two point each.

   (a) [1 points] True or False. Die cost is proportional to die area^2:

   (b) [1 points] True or False. CPU chip cost is the dominant cost in workstation system hardware cost:

   (c) [1 points] What instruction is used for subtract immediate on the MIPS?

   (d) [1 points] How would you execute a NOP operation on a MIPS processor?

   (e) [1 points] The MIPS uses register 31 for a "link register" in function calls. What instruction is used for "return"?

   (f) [2 points] In 20 words or less, explain why the JAVA virtual machine is a stack based architecture.

   (g) [2 points] A carry-lookahead adder can complete an N-bit add in time proportional to log(N) and requires a number of gates proportional to (choose one): log(N), N, N^2.

   (h) [1 points] Given the following MIPS-like program, with memory locations `a`,`b`, and `c`:

```
lw    r1,a
lw    r2,b
add   r3,r1,r2
sw    r3,c
```

   Write the same program for a stack machine:

   (i) [1 points] Write a short program for a memory–memory machine for the same operation:

(j) [1 points] What is the IEEE FP representation for 0?

(k) [2 points] What is the IEEE FP representation for 1.0×2^{-127}?

(l) [1 points] Which operation is more complex in IEEE FP, addition or multiplication?

(m) [2 points] Write the MIPS instruction representation (in 1's and 0's) for "ADD \$1,\$2,\$3". (Hint: For ADD, func=100000)

(n) [1 points] Write a formula for MIPS (millions of instructions per section) in terms of average CPI.

(o) [2 points] List two methods that different ISAs use for evaluating branch conditions.

2. **[20 points]** Consider the design of a 12-bit processor with the following ISA:

   (a) 12-bit instructions and 12-bit datawords.

   (b) *Word* addressing (memory address refer to words *not bytes*).

   (c) A single instruction format, with two register specifiers, and no immediates:

OPCODE (4 bits)	RA (4 bits)	RB (4 bits)

   (d) 16 general purpose registers.

   (e) The following set of instructions:

Instruction	Name	RTL description
Arithmetic Add	ADD RA,RB	Regfile[RA] $\leftarrow$ Regfile[RA] + Regfile[RB]
Arithmetic Sub	SUB RA,RB	Regfile[RA] $\leftarrow$ Regfile[RA] - Regfile[RB]
Logical OR	OR RA,RB	Regfile[RA] $\leftarrow$ Regfile[RA] OR Regfile[RB]
Load word	LDW RA,RB	Regfile[RA] $\leftarrow$ DataMemory[RB]
Store word	STW RA,RB	DataMemory[RB] $\leftarrow$ Regfile[RA]
Branch if Equal Zero	BEQ RA,RB	IF Regfile[RA] == 0 THEN PC $\leftarrow$ Regfile[RB]

Your task is to design a **single cycle processor**.

(a) Specify the type of components that you will need to build the datapath. Draw a symbol for each type and clearly label the inputs and outputs of each. Give each a name to signify its function.

(b) Connect the components into a datapath and draw all connections. Clearly label all the control signals and **list them again** below your datapath. Don't forget the program counter. Do not design the controller.

Control signals:

3. [15 points]

Consider a *single-cycle* processor with a set of instructions shown in the table below. Along with each instruction is its associated execution frequencies for some program X. Also shown is the delay of the critical path through the processor for each instruction. Assume that program X executes for exactly 10,000 instructions. Also assume that the processor has a clock period of 100ns.

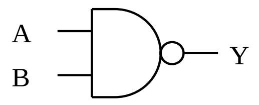
Instruction	Frequency	Delay
load	20%	100ns
store	10%	80ns
alu (r-type)	60%	45ns
branch	10%	45ns

(a) What is the CPU time for this processor on program X?

(b) Suppose now we are allowed to shorten the cycle time to 50ns and allow some instructions to take multiple cycles. (Of course, this change would require some modifications to the datapath and control, but don't worry about that.)

What is the CPU time for this new processor on program X?

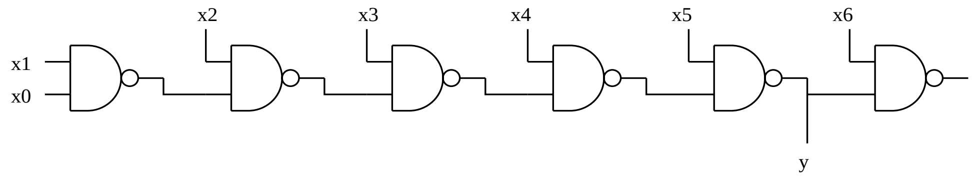
4. [15 points] Suppose that a NAND gate has the slope–intercept delay model shown below:



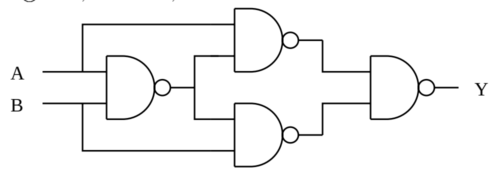
Input Capacitance on A and B = 100fF
Internal delay (intercept) = 1ns (for all transistions)
Output "slope" = 0.01 ns/fF (for all transistions)

(a) In the circuit shown below, what is the delay from input x_1 to y, assuming that all inputs are stable at time $t = 0$?

In this part and the other parts below, ignore delay due to wires.



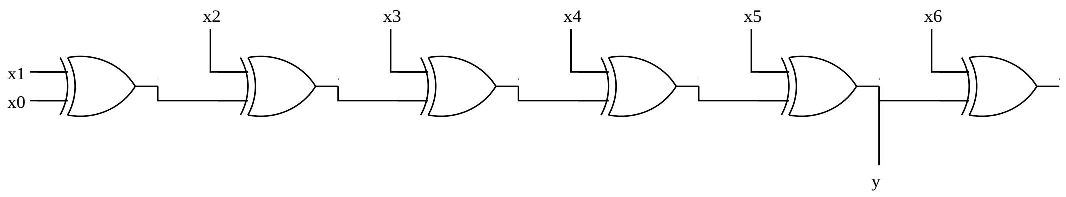
(b) Suppose we construct a new gate, XOR, as follows:



Fill in the table to define its slope–intercept delay model:

Model Parameter	Value
A input capacitance	
B input capacitance	
Internal Delay	
Output Slope	

(c) In the circuit shown below using our new gate, what is the delay from input x_1 to y, assuming that all inputs are stable at time $t = 0$?



5. [30 points]

   Recall that in class we looked at two designs for unsigned N-bit multiplication. The first design works without a clock signal and uses N^2 full-adder cells, organized as N N-bit adders. This is called an *array multiplier* or a combinatorial multiplier. The second multiplier design uses N full-adder cells, organized as an N-bit adder, and takes N clock cycles. This design, called a *shift and add multiplier*, reuses the N-bit adder N times to arrive at the result. Temporary results and operands are stored in registers (some with the ability to shift).

   Consider the design of a third type of multiplier. This one uses only one full-adder cell, organized as a 1-bit adder and takes N^2 clock cycles to multiply 2 N-bit unsigned integers.

   (a) Draw a block diagram for such a design using the following building blocks: shift registers , 1 full-adder cell, 1 or more flip-flops, and any simple logic gates you may need. The shift registers can be of any size needed, can shift in either direction, have all internal bits available as outputs, and can accept a new bit to be shifted in. You may assume that shift registers are automatically initialized with any value you want, but state these assumptions in part b). There are many ways to design this multiplier, but more credit will be given to simplier designs and those that try to minimize the amount of state needed.

   Draw the multiplier on the back of another page first and the transfer your final drawing to the answer space. Draw neatly!

(b) Write the sequence of steps the controller must generate (as we did for the shift and add multiplier in class):