CS152 Computer Architecture and Engineering
Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

Sp97 D.K. Jeong

Midterm #1
March 3, 1997

Name	Key
SID Number	
Discussion Section	

You may bring one double-sided note. You have 180 minutes. Please write your name on this cover and also at the top left of each page. The point value of each question is indicated in brackets after it. Make sure to show your work to get at least partial credit.

Problem	Possible	Score
1	15	
2	20	
3	10	
4	20	
5	15	

Total	80	

Name:

Question 1. (Performance) [15 pts]

You are to improve cost/performance on an existing system based on a single-chip microprocessor having the following parameters.

Base Machine	Clock Frequency	100MHz	
	Die Size	10 mm x 10 mm	
	Instruction Mix	int	62 %
		FP	38 %
	CPI	int	1.6
		FP	4.2

(a) What is the MIPS number of the base machine? [2 pts]

CPI = Σ CPI_i * IC_I, where CPI is cycles per instruction, and IC is instruction count.

CPI = 1.6*.62 + 4.2 * .38 = 2.588

MIPS = Clk freq/CPI = 100 MHz/2.588 = <u>38.64 MIPS</u>

By using an optimizing compiler, the number of FP instructions is reduced by 20% and int instructions by 10% for the same application program.

(b) What is the new MIPS number of the base machine when using the optimizing compiler? (Instruction mix is changed!)[2 pts]

Instruction mix has changed, so that there is a different proportion of FP and int instructions. Therefore, we need to recalculate CPI:

CPI = (0.9 *.62*1.6 + .8*.38*4.2) / (.9*.62 + .8*.38) = 2.5169

The denominator represents the decrease in Instruction Count, due to the optimizing compiler.

MIPS_comp = Clk freq/CPI = 100MHz/2.5169 = <u>39.73 MIPS</u>

(c) When running the same application, how much performance gain do you expect from using such an optimizing compiler? [2 pts]

MIPS is about the same, but IC has reduced. The relative performance is given by:

$$\frac{\text{Performance\_comp}}{\text{Performance\_base}} = \frac{38.64 \text{ MIPS}}{39.73 \text{ MIPS} / (.9*.62 + .8*.38)} = \underline{1.128}$$

The optimizing compiler improves performance by 12.8%.

By re-designing an FP hardware, die size will be increased by 20%, its FP CPI will be reduced to 2.8, clock frequency will remain as the same.

(d) What is the new MIPS number of this improved machine with an ordinary compiler? [2 pts]

CPI = 1.6*.62 + .38*2.8 = 2.056

MIPS = 100 MHz/2.056 = <u>48.64 MIPS</u>

(e) Assuming die yield is proportional to the inverse of the cube of the die size, how much additional cost do you expect? Assume that the number of dies per wafer is inversely proportional to the die size and that all the cost is proportional to the die.[2 pts]

Cost is inversely proportional to yield—as yield goes down, cost goes up. The yield goes down by the cube of the area: if the area goes up by 1.2x, the yield goes down by $(1.2)^3$x . Also, we can fit less dies on a wafer as the area goes up. If the area increases by 1.2x, we can fit 1.2x less dies per wafer. Since the cost per wafer stays the same, the cost per die goes up.

$$\frac{Cost\_FP}{Cost\_base} = \frac{\dfrac{Cost/wafer}{dies/wafer * die\_yield}}{\dfrac{Cost/wafer}{dies/wafer * die\_yeld}} = \frac{\dfrac{1}{1/1.2 * 1/(1.2)^3}}{1} = (1.2)^4 = 2.073$$

The cost went up by 107%.

As an alternative, you are going use a new, scaled CMOS technology without re-designing. In that case, the clock frequency changes to 125 MHz, its wafer cost doubles, and its die size is reduced by 15%. Use the same yield rule.

(f) How much performance improvement over the base machine do you expect? [1 pts]

The CPI and IC is the same as the base machine. The only difference is the clock rate.

$$\frac{\text{Performance\_scaled}}{\text{Performance\_base}} = \frac{125\text{ MHz}}{100\text{ MHz}} = \underline{1.25\text{ improvement}}.$$

The scaled CMOS is 25% faster.

(g) How much additional cost over the base machine do you expect? [2 pts]

The new technology is 15% smaller, that is 1-.15 = .85x the base machine.

$$\frac{Cost\_scaled}{Cost\_base} = \frac{\frac{Cost/wafer}{dies/wafer * die\_yield}}{\frac{Cost/wafer}{dies/wafer * die\_yeld}} = \frac{\frac{2}{1/0.85*(1/0.85)^3}}{1} = 2*(.85)^4 = 1.044$$

.

The scaled CMOS costs 1.044 times as much or 4.4% more.

(h) Compare the two hardware options and determine which approach you will choose for better cost/performance ratio? [2 pts]

$$\frac{\frac{Cost\_scaled}{Perforamce\_scaled}}{\frac{Cost\_FP}{Performance\_FP}} = \frac{\frac{1.044}{1.25}}{\frac{2.073}{1.26}} = .50764$$

The cost/performance of the scaled CMOS is .507x that of the FP hardware, or 1.97 times better. So we would choose the scaled CMOS technology.

Name:

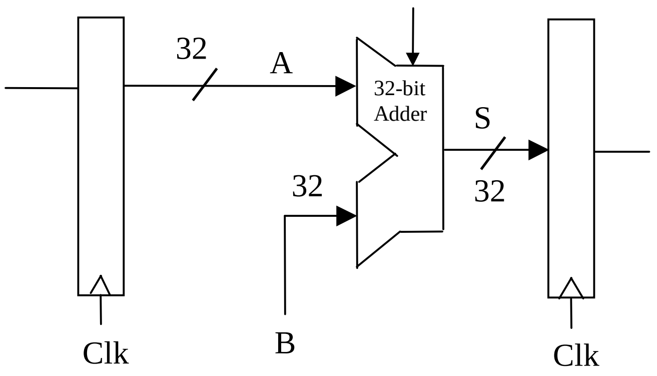
Question 2. (Delay) [20 pts]

You have the following gates and flip-flops available for your design. Do not assume any wiring capacitance in solving this problem. Try to minimize the overall delay.

	2-input NAND	2-input OR	2-input XOR	INV	2-input MUX
Input Load	100fF	100fF	100fF	50fF	100fF
TPhl, TPlh	0.3ns	0.4ns	0.5ns	0.2ns	0.3ns
TPhlf, TPlhf	0.001ns/fF	0.001ns/fF	0.001ns/fF	0.001ns/fF	0.001ns/fF

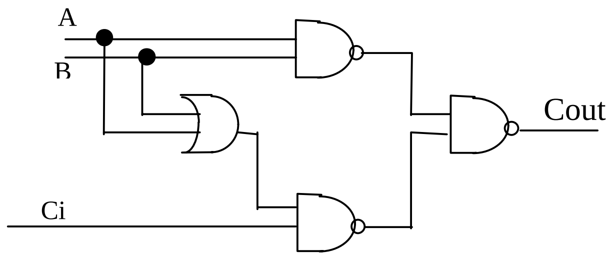
	D-FF
Set-up Time	2.0ns
Hold Time	1.0ns
Clock-to-Q Delay	1.0ns
TPhlf, TPlhf	0.001ns/fF

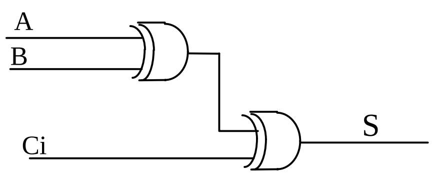
You are to design a circuit of the following logic block.



Name:

(a) Design a 1-bit adder (full adder) using only the above gates and characterize it. Try to minimize Cin -> Cout delay. [5 pts]





When calculating input loads, be sure to take into account fan-in. For example, A has an input load of 300 fF since it feeds three separate graphs.

Input loads: A=300 fF B=300 fF Ci = 200 fF

Output drives: Cout = .001 ns/fF S = .001 ns/fF

TPhl = TPlh for all cases. Must show all paths to fully characterize.

TP A,B → Cout = TP_OR + TPf_OR*C_{in}_NAND + TP_NAND + TPf_NAND* C_{in}_NAND + TP_NAND

TP A,B → Cout = .4 + .001*100 + .3 + .001*100 + .3

TP A,B → Cout = 1.2 ns

TP Ci→Cout = .3 + .001*100 + .3 = 0.7 ns

TP A,B→S = .5 + .001*100 + .5 = 1.1 ns

TP Ci→S = 0.5 ns

(b) Assuming no clock skew, what is the maximum clock frequency of the above block built with a 32-bit ripple carry adder using the circuit you design in (a)? [5 pts]

The longest path is not simply 32 times the carry chain path. Remember that the first input comes from the input FF and goes into A or B, and that the last output is through the 32nd sum bit to the output FF, not the 32nd carry bit.

So the critical path is input FF→A0 (or B0)→C0→C1→…→C30→S31→output FF.

Critical path = $T_{Clk\text{-}Q}$ + T_{su} + TPf_FF*C_{in}_A + TP_A→Cout + 30*(TPf_Cout*C_{in}_Ci + TP_Ci→Cout) + TPf_Cout*C_{in}_Ci + TP_Ci→S + TPf_S*C_{in}_FF + T_{hold}

Critical path = 1.0 + 2.0 + .001*300 + 1.2 + 30*(.001*200 + .7) + .5 + .001*100 + 1.0

Critical path = 33.3 ns

Cycle Time = 1/Critical path = 30 MHz

(c) What is the maximum clock skew allowed for correct operation with any input on B? [5 pts]

$T_{Clk\text{-}Q}$ + T_shortest path – T_{skew} > T_{hold}

The shortest path is from input FF to A or B to S to output FF, assuming no carries are generated or propagated.

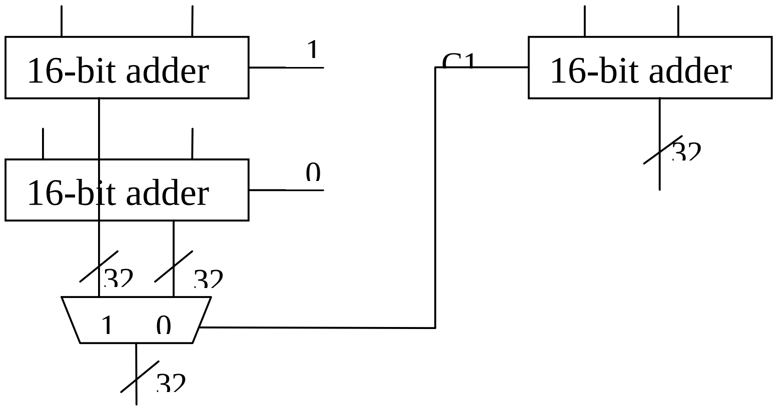
T_shortest path = TPf_FF*C_{in}_A + TP_A$\rightarrow$S + TPf_S*C_{in}_FF

= .001*300 + 1.1 + .001*100 =1.5 ns

1.0 ns + 1.5 ns – Tskew > 1.0 ns

<u>Tskew > 1.5 ns</u>

(d) Using full adders in (a) and muxes, sketch a carry select adder and estimate the maximum clock frequency. Do not use more than 48 such full adders. Assume no clock skew. [5 pts]



The critical path is calculated similarly to part b). We first calculate the delay from the input FF to A0 to C1…to C15. Then C15 must drive 16 muxes, which will be a significant delay, because it is a very big load. To optimize this, you might use inverters in parallel as buffers, since they would have a better TPf. Finally, the outputs from the muxes go to the output FF.

Critical path = $T_{Clk\text{-}Q}$ + T_{su} + TPf_FF*C_{in}_A + TP_A$\rightarrow$Cout + 15*(TPf_Cout*C_{in}_Ci + TP_Ci$\rightarrow$Cout) + TPf_Cout*16*C_{in}_mux + TP_mux + TPf_mux*C_{in}_FF + $T_{hold.}$

Critical path = 1.0 + 2.0 + .001*300 + 1.2 + 15*(.001*200 + .7) + .001*16*100 + .3 + .001*100 + 1.0

Critical Path = 21 ns

Cycle Time = 1/Critical Path = 47.6 MHz

By using inverters as buffers to drive the sixteen muxes we can reduce the critical path. Note that we can use only one inverter stage, and have our select signal be inverted, and just switch the inputs to the muxes, so that if C15 is one, it will be inverted but still select the correct input to the mux. We will use 4 inverters in parallel. Without buffers, it took 1.6 ns to drive the muxes. With 4 inverters in parallel, this is reduced to .8 ns (including the inverter delay, so that the .8 ns is cut off the critical path.

Name:

Question 3. (Floating-point) [10 pts]

(a) Add the two single precision FP numbers in IEEE format and represent the result in IEEE format. Show all the calculation and intermediate result. Use IEEE standard rounding method (nearest even in case of a tie) and make sure to include Guard, Round, and Sticky bits in calculation. [6 pts]

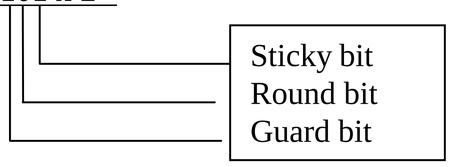
0 10011000 00000000000000000000000
0 10000000 00000000000000000000001

The exponent of the top number is given as $10011000_2 - 127_{10} = 152 - 127 = 25_{10}$
The exponent for the second number is 128 – 127 = 1.
This corresponds to:

$$\begin{array}{r} 1.000\ldots00 \times 2^{25} \\ \underline{+\ 1.000\ldots01 \times 2^{1}} \end{array}$$

We then shift the bottom number to the right 24 places, so that the radix points line up. As we shift, the sticky bit gets turned on, since the 1 that was originally the LSB of the significand gets shifted out to the right. The one to the left of the radix point will be shifted all the way out to the guard bit, one place to the right of the LSB of the significand.

$$\begin{array}{r} 1.000\ldots00\ 000 \times 2^{25} \\ \underline{+\ 0.000\ldots00\ 101 \times 2^{25}} \end{array}$$



The result is rounded up since the round bit corresponds to ½ and the sticky bit is on. The sum is $1.00000000000000000000001 \times 2^{25}$ =

0 10011000 00000000000000000000001

(b) Without Sticky bit, what would have been the result and why? [4 pts]

The guard bit tells us we are at ½. The round bit is zero and there is no sticky bit, so we must be at exactly ½. Since the LSB is zero, we would round down to the nearest even. Our sum would be exactly 1.0×2^{25}, which is different from our result of part (a).

0 10011000 00000000000000000000000

Name:

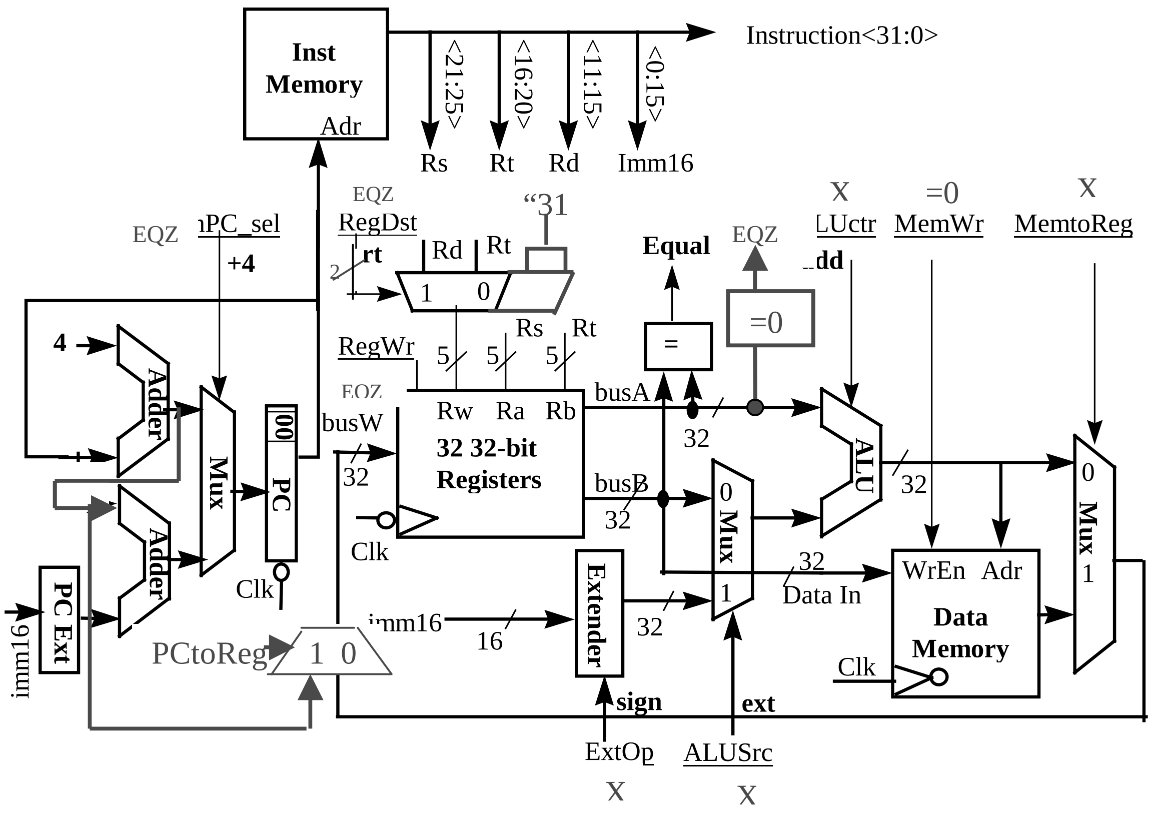
Question 4. (Single Cycle Datapath) [20 pts]

What changes and additions are needed to the datapath to support the following instructions? Make modifications on the single cycle datapath for MIPS from class. When you add control signals, put new names. Set proper values on all control signals (you may have to change exiting values and some are don't cares). Use red pen.

(a) A new instruction, BEZAL (branch on register being zero and link), which saves PC+4 in R31 and jump to the target in PC-relative addressing mode on the register, Rs, being zero (offset is with respect to PC+4).[10 pts]

BEZAL Rs, imm16

We need to add a component that checks if Reg[Ra] = 0. If it is, it produces the control signal EQZ, which is used to choose the branch address for the next PC, and to activate the write register. Another control signal PCtoReg writes PC+4 to the register file. Other control signals are set to 0,1 and X accordingly.

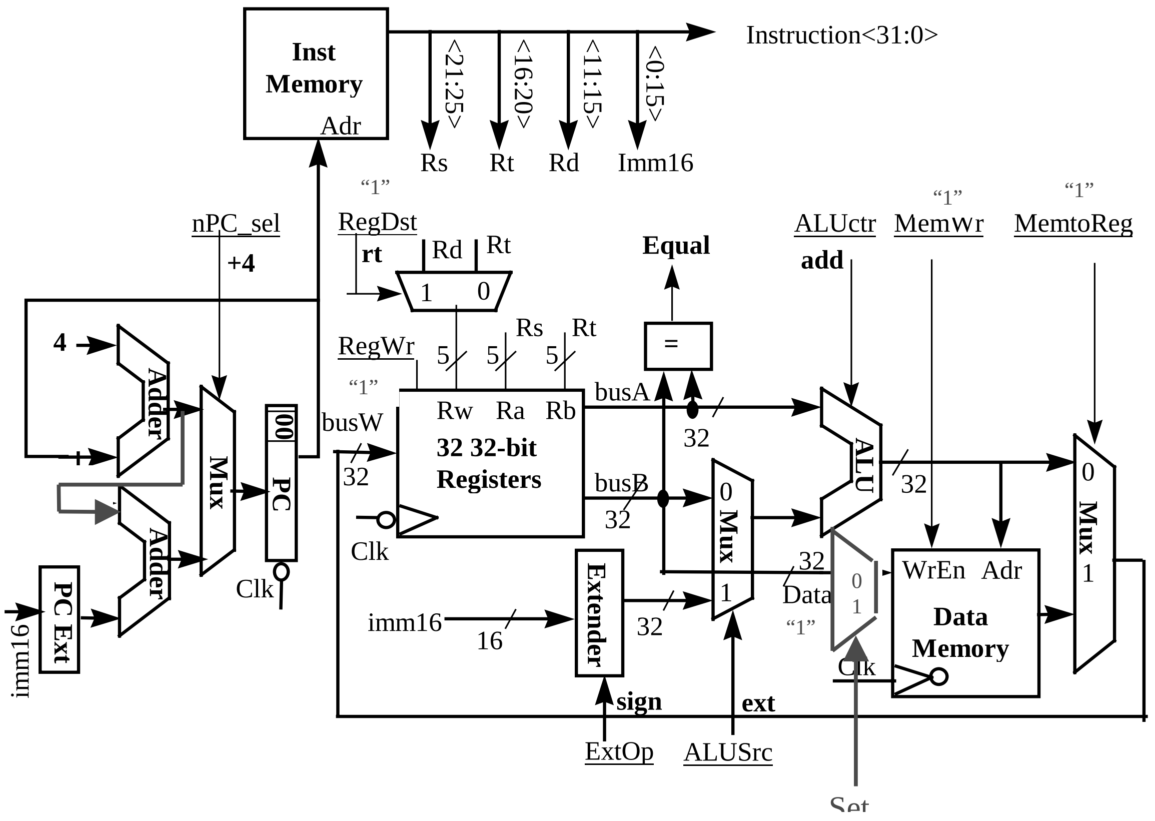


Name:

(b) An instruction, TAS (Test and Set), which loads memory to a register and stores 1 in the same memory location. Memory is addressed with the same addressing mode as a LOAD instruction. [10 pts]

TAS Rt, imm16(Rs)

We can do all reading and writing in the same cycle, since reads are simply combinational logic, and writes happen on the falling edge of the clock. Control signals are added to insure that the data from the memory is written to register Rt, the write enable of the memory is on, and a "1" will be muxed into the write data of the memory.



Name:

Question 5. (MIPS Assembly Language) Complete the optimized MIPS assembly language program for the following C program. Assume that delayed branches are used. Only one instruction per slot. Try to follow the register usage convention described in COD appendix A. Add appropriate comments to explain your idea. [15 pts]

```
int fibonacci (int k)
{
if (k == 0 || k == 1) return (1);
return (fibonacci (k-1) + fibonacci (k-2));
}
```

```
fibonacci:
     subu $sp,$sp,32
     sw   $31,24($sp)
     sw   $17,20($sp)
     move $17,$4
     sltu $2,$17,2            # check if k<2________________
     bne  $2,$0,$L2
     sw   $16,16($sp)____     # store saved register—will
                               have result of last fibonacci
     jal  fibonacci
     addu $4,$17,-1
     move $16,$2              # (this moves result of fib
                                 into $16, to be saved later

     jal  fibonacci
     addu $4,$17,-2______     # find fib(k-2).Remember this
                                is in the delay slot of jal

     j    $L3
     addu $2,$16,$2______     # $2 has fib(k-2), $16 has____
                                fib(k-1), add them together,
                              and return as fib(k-1)+fib(k-2)

$L2:
     li   $2,0x00000001       # 1
$L3:
     lw   $31,24($sp)____     # restore return address______
     lw   $17,20($sp)____     # restore saved registers_____
     lw   $16,16($sp)____     # ____"_____"______"__________
     addui$sp,$sp,32_____     # restore stack pointer_______
     j    $31
     .end fibonacci
```

(end of Midterm #1)