EE116C/CS115B Midterm

Total points: 100

Calculators, one double-sided handwritten sheet and MIPS cheatsheet from the book allowed.

Total time: 1.5 hours

Q1. 15points

A = X*Y + X*Z

Which of the following machines – stack-based, accumulator-based, register-memory based, or load-store based -- will implement this in maximum number of instructions? Which one in minimum? Explain. Assume ISA has ADD, MULT operations available.

Answer:

stack-based	accumulator-based	register-memory based	load-store based
push X	load X	mult T1,X,Y	load R1,X
push Y	mult Y	mult T2,X,Y	load R2,Y
mult	store T	add A ,T1,T2	mult R3,R1,R2
push X	load X		load R4,Z
push Z	mult Z		mult R5,R1,R4
mult	add T		add R6,R3,R5
add	store A		store C,R6
pop A			

So register-memory based will implement this in minimum number of instructions, while stack-based will implement this in maximun. Because stack-based can handle only one register at a time, or say one operand, while register-memory based can handle three registers or memory location at a time.

Q2. 20points

Here are two bit strings in hexadecimal:

0x7f6cbcef and 0xff70e1ab

Will overflow occur when you add them up if they are:

a) signed integers in two's complement?

b) single-precision IEEE floating point?

For each case, give reasons why or why not.

Answer:

0x7f6cbcef=0111 1111 0110 1100 1011 1100 1110 1111_{two}

0xff70e1ab=1111 1111 0111 0000 1110 0001 1010 1011_{two}

For both a) and b), overflow won't occur, since the sign bit of 0x7f6cbcef and 0xff70e1ab are different.

Q3. 25points

We want to add a new MIPS instruction so that the following C statement (p is a pointer to an int, and CONSTANT is small and can be negative) could be performed in one I-type MIPS instruction: *p = CONSTANT.

a) Make up the syntax for the I-typeMIPS instruction (call it sc for "store constant") that does it (show an example if the pointer lives in $v0 and the CONSTANT is 42).

Answer:

OP	rs		immediate

sc $v0, 42

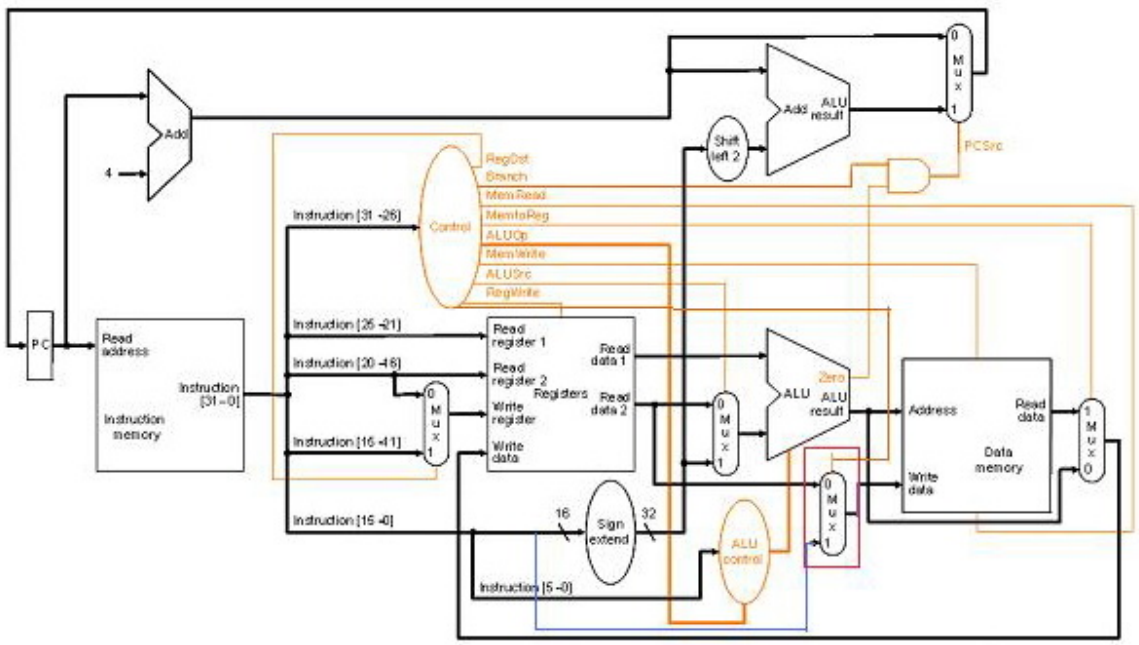
b) Below is the single-cycle MIPS datapath presented during lecture. Your job is to modify the diagram to accommodate a new MIPS instruction. Your modification may use simple adders, shifters, mux chips, wires, and new control signals. If necessary, you may replace original labels. **Modify the picture below** and list your changes.

Answer:

Changes:

1) One mux is needed between register file and memory. One of the input connects with read data2 of register file, and the other connects with instruction[16-0]. The output connects with write data of memory.

2) One control bit is needed to control the new added mux. When the sc is performed, the control bit is set as 1 to choose instruction [16-0]. Otherwise, the control bit is set as 0.



Q4. 40points

a) Suppose that 30% of the instructions are loads and half of all load instructions are followed by a dependent instruction. If this hazard creates a single-cycle delay, how much faster is the ideal pipeline machine (with a CPI of 1) that does not delay the pipeline than the real pipeline?

Answer:

Effective CPI for real pipeline machine= 70%X1+30%X(50%X1+50%X2)=1.15

so the idea pipeline machine is 1.15 times faster than the real pipeline.

b)Where are no-ops needed to avoid stall in a classic 5-stage pipeline for the following piece of code. Assume no data forwarding except register file bypassing.

sub $2, $1,$3

and $4, $2,$5

or $8, $2,$6

add $9, $4,$2

slt $1, $6,$7

Answer:

sub $2, $1,$3

no-op

no-op

and $4, $2,$5

or $8, $2,$6

no-op

add $9, $4,$2

slt $1, $6,$7

c) For a classic 5-stage pipeline that does not have any kind of branch prediction, three clock cycles are wasted for every branch.

Branch instruction IF ID EX MEM WB

Branch successor IF stall stall IF ID EX MEM WB

If branch frequency is 30% and ideal CPI (without any stalls) is 1, what is the CPI considering branch stalls ?

Answer:

30%X4+70%X1=1.9

d) For some hypothetical pipeline, it takes three pipeline stages before the branch target address is known and additional cycle before the branch condition is evaluated. What would be branch penalities for the following simplest prediction schemes for the three kinds of dynamic branches :

Answer:

Branch Scheme	Penalty Unconditional	Penalty untaken	Penalty taken
Flush Pipeline	3	4	4
Predict taken	3	4	0
Predict untaken	3	0	4

e) For some program, the only branches are FOR loops. Moreover all FOR loops are of form:

for(i=0;i!=3;i++){…}

Can you suggest a simple branch prediction strategy which will give zero mispredictions stating any assumptions you have made ?

Answer:

T-T-T-N