Name:_____

CSE 30321 – Computer Architecture I – Fall 2009
Midterm Exam
October 15, 2009

Test Guidelines:

1. Place your name on ***EACH*** page of the test in the space provided.
2. Answer every question in the space provided. If separate sheets are needed, make sure to include your name and clearly identify the problem being solved.
3. Read each question carefully. Ask questions if anything needs to be clarified.
4. The exam is open book and open notes.
5. All other points of the ND Honor Code apply. By writing your name on the exam, you agree to abide by the ND Honor Code.
6. Upon completion, please turn in the test and any scratch paper that you used.

Suggestion:

- Whenever possible, show your work and your thought process. This will make it easier for us to give you partial credit.

Question	Possible Points	Your Points
1	30	
2	15	
3	15	
4	20	
5	20	
Total	100	

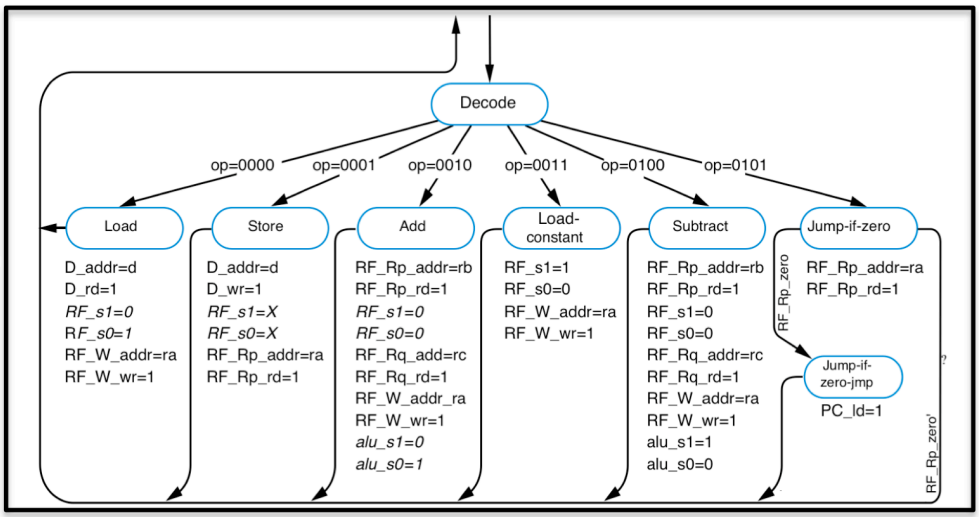
Name:_____

Problem 1: (30 points)

The following question is based on the extended Vahid processor that you have worked with in class and/or lab. For your convenience, (i) the instructions, (ii) their opcodes, and (iii) the finite state machine (FSM) are all shown below. **You can assume that R0 is *always equal to 0*.** Also – there is no need to take "wait states" into account as was done in the lab.

Instruction	Meaning
MOV Ra, d	RF[a] = D[d]
MOV d, Ra	D[d] = RF[a]
ADD Ra, Rb, Rc	RF[a] = RF[b]+RF[c]
MOV Ra, #C	RF[a] = C
SUB Ra, Rb, Rc	RF[a] = RF[b]-RF[c]
JMPZ Ra, offset	PC=PC+offset if RF[a]=0

Instruction	Opcode
MOV Ra, d	0000
MOV d, Ra	0001
ADD Ra, Rb, Rc	0010
MOV Ra, #C	0011
SUB Ra, Rb, Rc	0100
JMPZ Ra, offset	0101



Question A(i): (10 points)

Write a sequence of Vahid instructions that is equivalent to the following pseudo code. Be sure to add a comment by each line of your code so that we can follow your thought process and give partial credit where possible.

```
i = 2;                    // i maps to R2
j = 3;                    // j maps to R3

k = i + j;                // k maps to R5

if (i == 6)
      x = D[1];           // x maps to R8
else
      x = D[2];           // x maps to R8

y = D[3];                 // y maps to R7
```

Name:_____

Question A(ii): (5 points)
Given the data supplied in Part A(i), what is the CPI for your code?

Name:_____

Question B(i): (10 points)

Explain what the sequence of Vahid instructions in the table below ultimately accomplishes.

- Note 1:
  - Psuedo-code is fine, just don't translate instruction-by-instruction. Instead, summarize at some higher level (e.g. "while (n < 17) …")
- Note 2:
  - This code includes a "jump-if-negative" instruction – where the PC will change if the register value is negative. It is executed just like the "jump if zero" instruction.
- Note 3:
  - There is space in the table below for comments. I will look at anything you write here when considering partial credit.

Address	Label	Instruction / Data	Comment
…		…	
10		mov R5, $10_{constant}$	
11		mov R6, $6_{constant}$	
12		mov R8, $8_{constant}$	
13		sub R7, R5, R6	
14		jumpn R7, end	
15		mov R2, $18_{address}$	
16		add R3, R2, R8	
17		mov $18_{address}$, R3	
18	end:	mov R5, $30_{address}$	
…		…	
30		data word 1	
31		data word 2	
32		data word 3	
…		…	
N		data word N	

Question B(ii): (5 points)

In what clock cycle(s) does R5 change state? (i.e. when does the value of R5 change?) Assume the instruction at address 10 is fetched in clock cycle 1.

Name:____________________________

Problem 2: (15 points)

Translate the following pseudo code to MIPS assembly. Please use the table below – and again, add comments so that we can give you partial credit if needed.

```
if (x == 1) {
      j = 4;
}
else if (x == 2) {
      j = 7;
}
else {
      j = 11;
}
array(j) = b(3)
```

Assume the following:
- The starting address of b(0) is stored in $3
- The starting address of array(0) is stored in $2
- Assume x maps to $12
- Assume j maps to $22

Note: You do not need to use all of the rows.

Label:	Instruction:	Comment:

Name:_____

Extra table:

Label:	Instruction:	Comment:

Name:_____

Problem 3: (15 points)

For this question, assume that the processor you are working with executes different instruction types in integer numbers of short clock cycles (i.e. a multi-cycle approach). The machine's clock rate is 1 GHz and 50,000,000 instructions are executed. The profile of a common workload is shown below:

	Instruction Type	Percentage	Number of CCs
A	Load	20%	6
B	Store	10%	5
C	Integer ALU	15%	5
D	Floating Point ALU	30%	10
E	Branch	15%	5
F	Jump	10%	4

Question A: (10 points)

You are considering 3 potential improvements to improve the performance of this workload.

1. You can spend time improving the compiler. This will reduce the total instruction count from 50,000,000 to 40,000,000. You can assume that the above percentages will still hold.
2. You can improve the clock rate from 1 GHz to 1.25 GHz. However, each instruction will take 1 CC longer to execute.
3. You can keep the instruction count and the clock rate the same, but reduce the number CCs per floating point instruction from 10 to 5.

Which option is best?

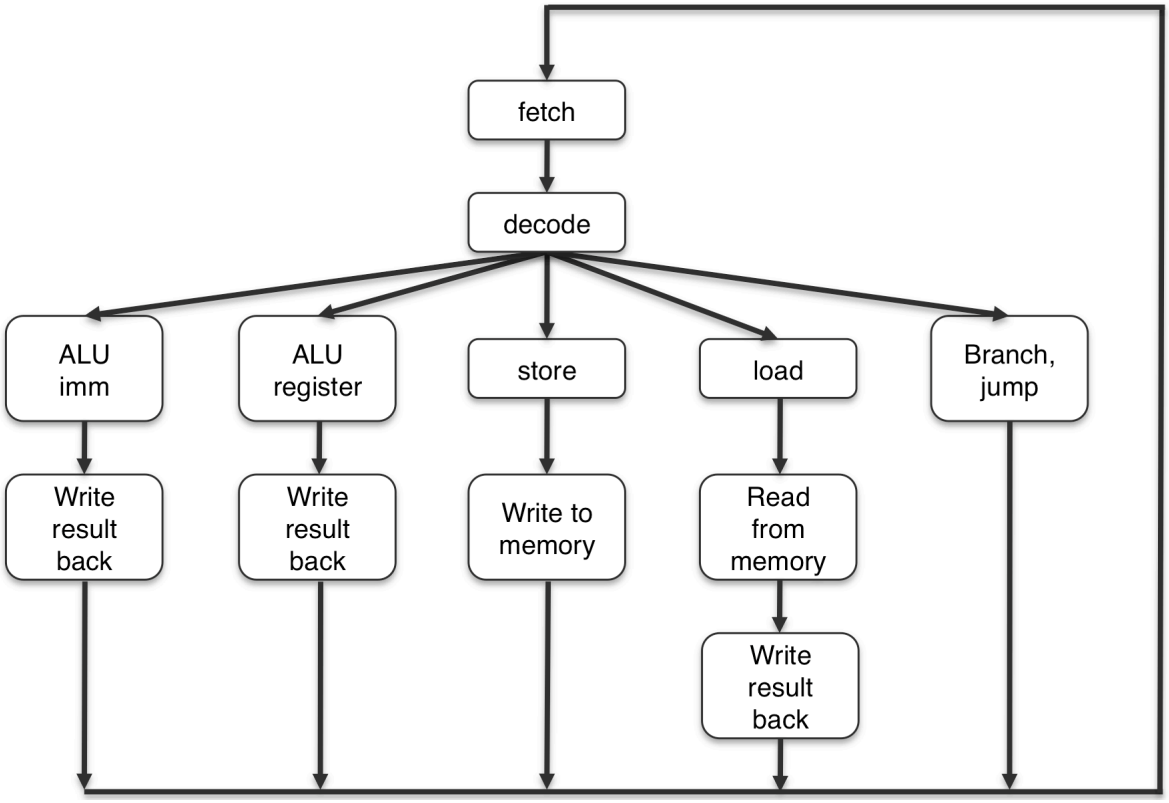
Question B: (5 points)

What is the speedup over the base case?

Name:_____

Problem 4: (20 points)

A simplified version of a MIPS finite state machine (for a multi-cycle processor) is shown below:



The clock rate for this machine is 1 GHz.

Question A: (10 points)

How long will it take to execute the for loop instruction sequence shown below?

assume $10 = 40 and $5 = 0

```
loop:   lw      $4, 0($5)
        sll     $4, $4, 2
        sw      $4, 0($5)
        addi    $5, $5, 4
        bne     $5, $10, loop
end:
```

Name:_____

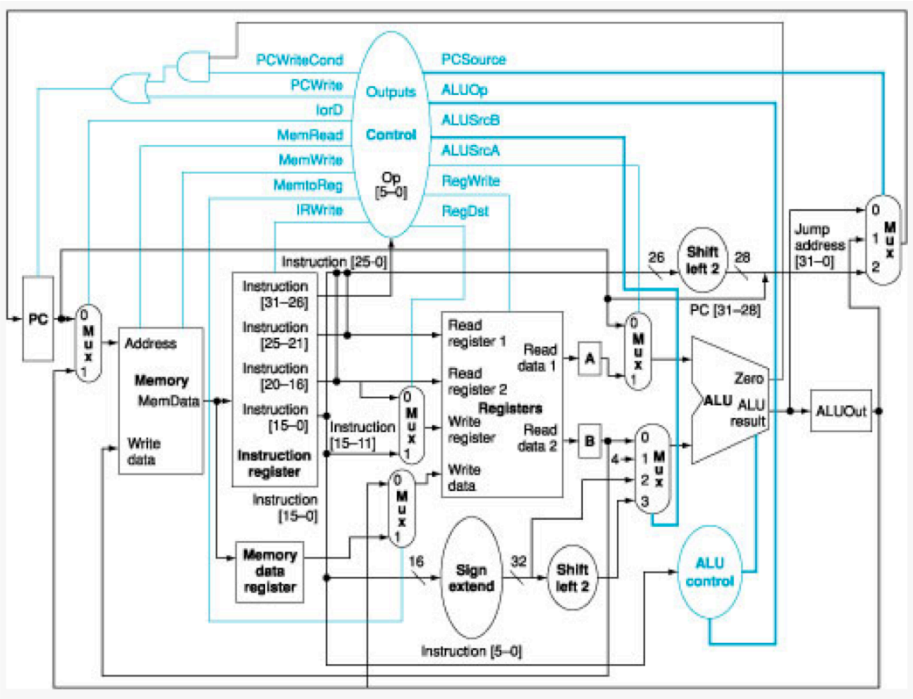
Question B: (10 points)
Assume that you can add a new instruction called "store word auto-increment" which also takes 4 CCs. This instruction is the same as the normal "store" instruction that you have used to date, except that by the end of the instruction, the register used to calculate the address is incremented by 4. Thus:

Assembly: sw-ai $5, 0($4)
RTL: $5 ← memory(0 + contents of $4) ll $4 ← $4 + 4

Note, the value of $4 used in the address calculation is the previous value of $4, not $4 + 4.

While this instruction will reduce the number of instructions in the above loop, because the new instruction will need to do more computation, the clock period could increase. How much of a "slow down" can we tolerate for the above code if we used the new instruction?

Bonus Question: (3 points)
Will the MIPS multi-cycle datapath need to be modified to support this new instruction? If so, how?



Name:_____

Problem 5: (20 points)

Question A: (5 points)

Consider the following snippet of code:

```
int main(void) {
      int A = 1;       // A maps to $ t1
      int B = 2;       // B maps to $ s1
      int C = 3:       // C maps to $ s2

      ….

      printf("B is equal to %d.", B);

      B = C + C;
}
```

When the printf() function is called, per the MIPS calling convention, what registers are saved and where are they saved? Briefly justify your answer for full/partial credit.

Question B: (5 points)

Consider the slightly modified version of the above code shown below. What registers ***need*** to be saved to the stack? Per the MIPS calling convention, how would this be accomplished?

```
int do_anything();

int main(void) {
      int A = 1;       // A maps to $ t1
      int B = 1;       // B maps to $ s1
      int C = 2;       // C maps to $ s2
      int D = 2;       // D maps to $ t2
      int X;

      …

      X = do_anything();

      B = C + C;
      A = A + D + B;
}
```

Name:_____

Question C: (5 points)
Consider the code shown below. Assuming that no other functions are called in this sequence, what is the value of $s1 (a) immediately before the call to "do_something2()" and (b) immediately before the statement "X = Y – Z"?

```
int main(void) {
      int B = 1;        // B maps to $ s1
      int C = 2;        // C maps to $ s2

      …

      do_something1();

      B = B + C;
}

int do_something1() {
      int P = 3;        // P maps to $ s3
      int Q = 4;        // Q maps to $ s4
      int R             // R maps to $ s5

      R = P + Q;

      do_something2();
}

int do_something2() {
      int X = 6;        // X maps to $ s6
      int Y = 7;        // Y maps to $ s7
      int Z             // Z maps to $ s8

      X = Y - Z;
}
```

Question D: (5 points)
Assume $sp initially = 200. What is the value of the $sp if 3 function calls are made and 3 data words must be saved to the stack for each call?