FACULTY OF ENGINEERING AND ARCHITECTURE **SPRING 2006**

AMERICAN UNIVERSITY OF BEIRUT

EECE 321 – COMPUTER ORGANIZATION

PROF. MANSOUR **MIDTERM I (HW 8)** MARCH 27, 2006

NAME: ________________________ ID: __________________________

INSTRUCTIONS:
• **CLOSED BOOK/CLOSED NOTES**
• **EXAM DURATION: 2 HOURS**
• **WRITE YOUR NAME AND ID NUMBER IN THE SPACE PROVIDED ABOVE.**
• **WRITE YOUR ANSWERS ON THE QUESTION SHEET.**
• **THE SCRATCH BOOKLET WILL NOT BE CONSIDERED IN GRADING.**
• **WRITE COMMENTS NEXT TO YOUR MIPS INSTRUCTIONS.**
• **BE AS CLEAR AND AS NEAT AS POSSIBLE.**
• **WRITE DOWN ANY ASSUMPTIONS YOU USE IN SOLVING ANY PROBLEM.**

PROBLEM	MAX POINTS	GRADE
1	12	
2	24	
3	12	
4	22	
5	12	
6	22	
7	16	
8	20	
TOTAL	140	

Problem 1: Single Cycle MIPS Datapath [12 points]

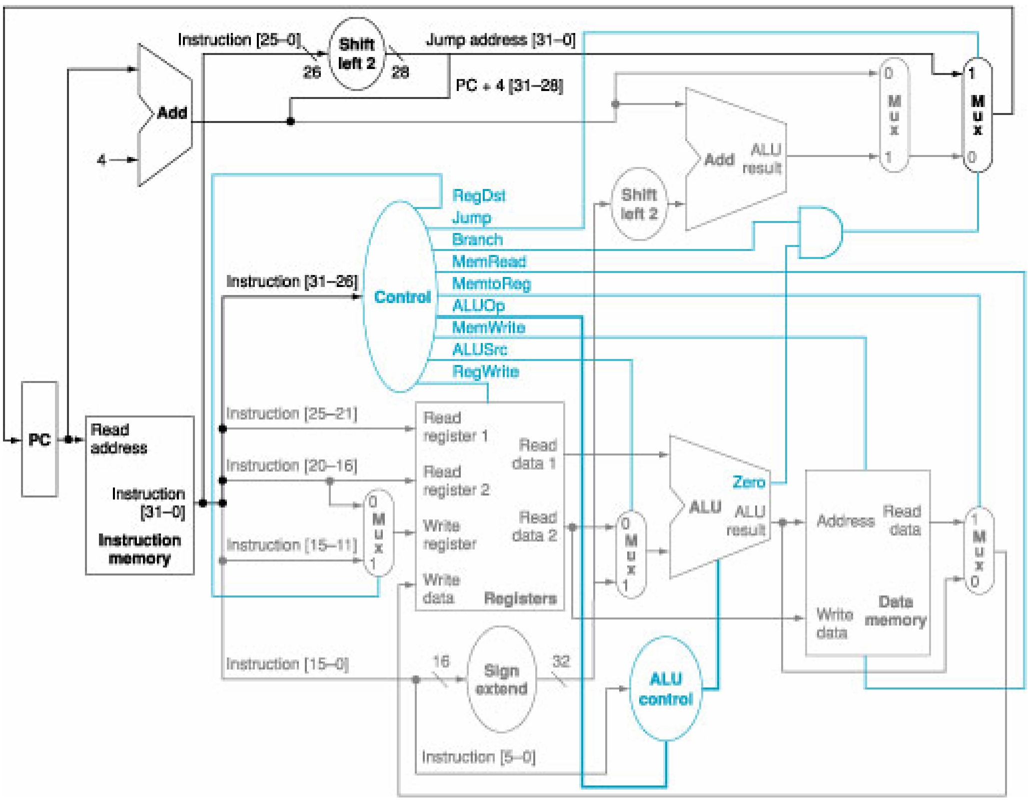
Modify the MIPS single-cycle datapath shown below to support the following instruction:

`jral L`

which performs a jump-and-link and saves the address of the next instruction after **`jral`** in **`$ra.`** You are allowed to modify the existing functional units and/or add extra connections. To receive full credit, all your modifications should be clearly indicated on the figure below. Register **`$ra`** has number $31_{(10)}$, and the format of **`jral`** is: [**6 points**]

What control signals need to be added? [**3 points**]
Answer: ____

What combinational logic blocks need to be added? [**3 points**]
Answer: ____



Problem 2: MIPS Compilation [24 points]

Compile the following C++ statements shown on the left column. Assume the following register-to-variable mappings: **`i:$s0, j:$s1, A:$s2`**. You can only use the following MIPS instructions: **`sll, srl, add, addi, sub, lw, sw, slt, beq, bne, j`**. NO OTHER INSTRUCTIONS ARE ALLOWED. The variables **`i`**, **`j`**, and the array **`A`** are all integers.

C++ statement	MIPS assembly code
`A[i*64+j] = A[i+j*64];` **[5 points]**	
`*i = *j;` **[2 points]**	
`if(i>j) i = i + 1;` **[3 points]**	
`while(i<=j) i++;` **[4 points]**	
`for(i=0;i<10 && j>0;i++) A[j--] = i;` **[5 points]**	
`j = (`$2^{16}+2^{-16}$**`)*i;`** **[5 points]**	

<u>Problem 3</u>: MIPS Assembly **[12 points]**

Consider the following MIPS assembly code.

```
main:   addi $t0, $zero, 5
         add $t1, $zero, $t0
         addi $t2, $zero, 1
         add $t3, $zero, $t0
loop:   sub $t1, $t1, $t2
         beq $t1, $zero, finish
         add $t3, $t3, $t1
         j loop
finish:
```

a) What is the value of **`$t1`** when the program reaches "**`finish`**" (in decimal)? **[3 points]**

Answer: **`$t1 =`**

b) What is the value of **`$t3`** when the program reaches "**`finish`**" (in decimal)? **[3 points]**

Answer: **`$t3 =`**

c) Assume that add and sub require 1 instruction cycle, and **`j`** and **`bne`** require 2 instruction cycles, how many cycles does this code run when reaching "**`finish`**" (do not count "**`finish`**")? **[6 points]**

Answer: **`Cycles =`**

Problem 4: Floating Point [22 points]

a) You are writing a program that requires several floating point multiplications by $-2.5_{(10)}$. You decide to load this constant into the floating point register **`$f2`** using the following instruction:

```
mtc1 $t0, $f2     # $f2 ← $t0
```

However, before you can transfer the constant to **`$f2`** you have to load it into **`$t0`**. Write the shortest sequence of instructions that will load the constant $-2.5_{(10)}$ represented in the IEEE 754 single floating point standard into $**`t0`**. [6 **points]**

Answer:

b) Find a number **`x`** so that **`x + 1.0 = x`**, according to single precision floating-point arithmetic. **[4 points]**

Answer: **`x`** =

c) Determine what **`0x3e000000`** is in decimal, if we assume it represents an IEEE single precision number. **[6 points]**

Answer:

d) Assume you have a 64x32 matrix **`A`** of double precision *complex* numbers. Each element of the matrix is a complex number, with the real and imaginary parts are represented as double precision numbers. The matrix is stored in column major format, i.e., all entries in a column are stored sequentially in memory before storing the entries of the next column. The real part of each entry is stored first as a double number, followed immediately by the imaginary part, again stored as a double number. If the array is stored starting at byte address **`0x00400008`**, at what byte address is the array element **`A[3][6]`** stored? Write your answer in hexadecimal. **[6 points]**

Answer:

Problem 5: Pseudo Instructions [12 points]

a) Write a minimal-length sequence of MIPS instructions that implements the following pseudo-instruction. You are not allowed to use any **undefined** pseudo-instructions in this problem. Use the assembler register where necessary. **[6 points]**

`lw $s1,BIG($s0)` # like a normal load, but **`BIG`** may be very large and may need # more than 16 bits.

Solution:

b) Write a minimal-length sequence of MIPS instructions that implements the following pseudo-instruction:

`ble $s1,$s2,$s3`

which branches to address contained **`$s1`** if **`$s2`** is less than or equal to **`$s3`**. You are not allowed to use any **undefined** pseudo-instructions in this problem. Use the assembler register where necessary. **[6 points]**

Solution:

Problem 6: CPU Performance [22 points]

Consider the following C++ code. Assume you generated MIPS assembly code for the C++ code shown using two different compilers. Compiler 1 generated the code in Figure P6(a) while compiler 2 generated the code in Figure P6(b). For the scope of this question only, you should assume that all the instructions used by the compilers are real instructions (as opposed to pseudo-instructions) in the MIPS architecture. Assume arithmetic/logic instructions require 2 cycles to execute, loads 5 cycles, branches 3 cycles, and jumps 1 cycle.

```
int foo(int v[ ], int k, int n, int m){
      sum = 0;
      for(i=k ; i<n ; i=i+m)
            sum = sum + v[i];
      return sum;
}
```

Figure P6(a): Code generated by Compiler 1	Figure P6(b) : Code generated by Compiler 2

<pre>   add $t0, $zero, $zero
        add $t1, $a1, $zero
loop: bge $t1, $a2, return
        sll $t2, $t1, 2
        add $t2, $t2, $a0
        lw $t3, 0($t2)
        add $t0, $t0, $t3
        add $t1, $t1, $a3
        j loop
return: add $v0, $t0, $zero
        jal $ra</pre>
	<pre> add $v0, $zero, $zero
	        sll $a1, $a1, 2
	        sll $a2, $a2, 2
	        sll $a3, $a3, 2
	        add $t1, $a1, $zero
	        bge $t1, $a2, return
	loop: add $t2, $t1, $a0
	        lw $t3, 0($t2)
	        add $v0, $v0, $t3
	        add $t1, $t1, $a3
	        blt $t1, $a2 loop
	return: jal $ra</pre>

a) Fill the table below with the number of instructions of each type that will be executed by the code generated by compiler 1 and by compiler 2 when the function **`foo(v,0,100,1)`** is invoked. **[8 points]**

Instruction Type	Compiler 1 Code	Compiler 2 Code
Arithmetic/logic		
Loads		
Branches		
Jumps		

b) Compute the average number of clocks per instruction (CPI) for each version of the program. **[8 points]**

Solution:

CPI(*P1*) =

CPI(*P2*) =

c) Which version of **`foo()`**, P1 or P2, is faster if you run it on a 1 GHz processor? By how much? **[6 points]**

Solution:

<u>Problem 7</u>: Function Calls [16 points]

Assume the following MIPS program begins execution at **`0x00400008`**. **`main()`** is first invoked, which then calls **`subA`**, which calls **`subB`**. In the comments part of the code inside the boxes on the right, determine the values, memory location and its contents, that are changed <u>right after</u> the execution of each of those selected instructions. For example, for the instruction at **`0x00400020`**, what is the value in **`$sp`**, after its execution, etc. Assume that **`$sp = 0x7fffeffc`** at the start of the program. **`Syscall`** simply jumps and returns back to the next instruction after it. Fill in the 8 boxes below in hexadecimal.

```
Address              Instruction             Comments

0x00400008           sw $zero, 0($sp)        # mem[0x7fffeffc] = 0x0000000
0x0040000c           addi $v0, $zero, 0xf    # $v0 = 0x0000000f
0x00400014           jal main                # $ra = [          ]
0x00400018           ori $v0, $zero, 10      #
0x0040001c           syscall
0x00400020   main:   addi $sp $sp, -4        # $sp = [          ]
0X00400024           sw $ra, 0($sp)
0x00400028           jal subA                #
0x0040002c           lw $ra, 0($sp)          # $ra = [          ]
0x00400030           addi $sp, $sp, 4        #
0x00400034           jr $ra                  #
0x00400038 subA:     addi $sp $sp, -4        # $sp = [          ]
0X0040003c           sw $ra, 0($sp)          # mem[$sp] = [          ]
0x00400040           jal subB                #
0x00400044           lw $ra, 0($sp)          # $ra = [          ]
0x00400048           addi $sp, $sp, 4        # $sp = [          ]
0x0040004c           jr $ra                  # $ra = [          ]
0x00400050 subB:     jr $ra                  #
```

Problem 8: Branches and Labels [20 points]

a) If the instruction **`beq $1,$0,EXIT`** is located at address **`0x00400050`**, and encoded as **`0x10200007`**, what is the address in hexadecimal of the label **`EXIT`**? **[5 points]**

Answer:

b) The program below is written using the MIPS instruction set. It is loaded into memory at address **`0xF000000C`** (all instruction memory addresses are shown below). Assume the opcode field of the jump instruction is **`000010`**, write the machine instructions for the two jumps in the code below in hexadecimal. **[5 points]**

```
0xF000000C   Loop:    addi $1, $1, -1
0xF0000010            beq  $1, $0, ESCAPE
0xF000001C            beq  $2, $0, EXIT
0xF0000014            j    Loop
0xF0000018   ESCAPE:  addi $1, $1, 10
0xF000001C            addi $2, $2, -1
0xF0000020            j    Loop
0xF0000024   EXIT:
```

Answer:

c) What does the following assembly code compute? **[10 points]**

```
             add  $s1,$s2,$0
             jal  HERE
HERE:        slt  $s0,$s3,$s2
             sll  $s0,$s0,2
             addi $ra,$ra,20         # 20 is in decimal
             add  $ra,$ra,$s0
             jr   $ra
             add  $s1,$0,$s3
THERE:       …
```

Answer: