Brown University
School of Engineering
EN164 Design of Computing Systems
Professor Sherief Reda
HW 02. 140 points. Due Date: Tuesday, March 13th in class.

1. [15 points] Give three examples from the MIPS architecture of each of the architecture design principles: (1) simplicity favors regularity; (2) make the common case fast; (3) smaller is faster; and (4) good design demands good compromises. Explain how each of your examples exhibits the design principle.

2. [20 points - from Midterm 2011] You are given a processor with a stack-based architecture. The processor has only 4 assembly instructions:

```
PUSH   X      # pushes the contents of the memory location X into the top of
the stack
POP    X      # pops contents of the top of the stack and stores it in memory
location X
ADD           # pops the top two elements in the stack; adds them and pushes
the result into the top of the stack
SUB           # pops the top two elements in the stack; subtracts the top
element from the second top element and pushes the result into the top of the
stack
```

Write an assembly language program that would evaluate 5 - (2 + 1). Assume 1 is stored in memory location A, 2 is stored in memory location B, and 5 is stored in memory location C. Assume that the result should be written to memory location X. Make sure you program uses the minimum number of instructions.

3. [10 points] What is the range of instruction addresses to which conditional branches, such as `beq` and `bne`, can branch in MIPS? Give your answer in number of instructions relative to the conditional branch instruction.

4. [15 points] Convert the following program from machine language into MIPS assembly language. The numbers on the left are the instruction address in memory, and the numbers on the right give the instructions at that address. Then reverse engineer a high-level program that would compile into this assembly language routine and write it. Explain in words what the program does. `$a0` is the input, and it initially contains a positive number, `n`. `$v0` is the output.

[0x00400000]	0x20080000
[0x00400004]	0x20090001
[0x00400008]	0x0089502a
[0x0040000c]	0x15400003
[0x00400010]	0x01094020
[0x00400014]	0x21290002
[0x00400018]	0x08100002
[0x0040001c]	0x01001020

5. [20 points - from 1st Quiz 2011]

a. Write a MIPS assembly program to count the number of 1's in a 32-bit register. Assume the register is $s1, and the final count result should be in register $s2.
b. What is the execution time of your program if your MIPS processor frequency is 250 MHz and can execute 1 instruction per cycle?
c. What is the execution time of your program if your MIPS processor frequency is 200 MHz and can execute 2 instructions per cycle?

6. [15 points] Consider the following new instruction to include in the MIPS architecture:

```
LWI Rt, Rd(Rs)      #Reg[Rt] = Mem[Reg[Rd]+Reg[Rs]]
```

a. Which existing blocks (if any) can be used for these instructions?
b. Which new functional blocks (if any) do we need for this instruction?
c. What new signals do we need (if any) from the control unit to support this instruction?

7. [15 points] Assume the following latencies for logic blocks in the datapath:

```
I-MEM: 200 ps, Add: 70 ps, ALU: 90 ps, Regs: 90 ps, D-MEM: 250 ps, Sign-Extend: 15 ps, Shift-Left-2: 10 ps
```

a. What is the clock cycle time if the only types of instructions we need to support are ALU instructions (ADD, AND, etc.)?
b. What is the clock cycle time if we only have to support LW instructions?
c. What is the clock cycle time if we must support ADD, BEW, LW and SW instructions?

8. [15 points] Suppose that one of the following control signals in the signals in the single-cycle MIPS processors has a stuck-at-0 fault, meaning that the signal is always 0 regardless of its intended value. What instructions would malfunction? Why?

a) RegWrite
b) $ALUop_1$
c) MemWrite

9. [15 points] Many processor architectures have a load with post-increment instruction, which updates the index register to point to the next memory word after completing the load. `lwinc $rt, imm($rs)` is equivalent to the following two instructions:

```
lw $rt, imm($rs)
addi $rs, $rs, 4
```

Modify the single-cycle MIPS processor to implement the `lwinc` instructions. Sketch the datapath indicating the changes compared to the datapath given in the lecture, and name any control signals as well changes to the decoder.