(1) (10%) The following MIPS assembly program tries to copy words from the address in register $a0 to the address in $a1, counting the number of words copied in register $v0. The program stops copying when it finds a word equal to 0. You do not have to preserve the contents of registers $v1, $a0, and $a1. This terminating word should be copied but not counted.

```
loop:   lw      $v1, 0($a0)       # read next word from source
        addi    $v0, $v0, 1       # Increment count words copied
        sw      $v1, 0($a1)       # Write to destination
        addi    @a0, $a0, 1       # Advance pointer to next word
        addi    @a0, $a1, 1       # Advance pointer to next word
        bne     $v1, $zero, loop  # Loop if word copied != zero
```

There are multiple bugs in this MIPS program; fix them and turn in a bug-free version.

(2) (10%) Carry lookahead is often used to speed up the addition operation in ALU. For a 4-bit addition with carry lookahead, assuming the two 4-bit inputs are a3a2a1a0 and b3b2b1b0, and the carry-in is c0,

(a) (4%) first derive the recursive equations of carry-out ci+1 in terms of ai and bi and ci, where i = 0, 1, .., 3.

(b) (4%) Then by defining the *generate* (gi) and *propagate* (pi) signals, express c1, c2, c3, and c4 in terms of only gi's, pi's and c0.

(c) (2%) Estimate the speed up for this simple 4-bit carry lookahead adder over the 4-bit ripple carry adder (assuming each logic gate introduces T delay).

(3) (10%) When performing arithmetic addition and subtraction, overflow might occur. Fill in the blanks in the following table of overflow conditions for addition and subtraction.

Operation	Operand A	Operand B	Result indicating overflow
A + B	≧0	≧0	(a) (1%)
A + B	<0	<0	(b) (1%)
A - B	≧0	<0	(c) (1%)
A - B	<0	≧0	(d) (1%)

Prove that the overflow condition can be determined simply by checking to see if the CarryIn to the most significant bit of the result is not the same as the CarryOut of the most significant bit of the result (6%).

(4) (15%) Assume all memory addresses are translated to physical addresses before the cache is accessed. In this case, the cache is physically indexed and physically tagged. Also assume a TLB is used. (a) Under what circumstance can a memory reference encounter a TLB miss, a page table hit, and a cache miss? Briefly explain why. (b) To speed up cache accesses, a processor may index the cache with virtual addresses. This is called a *virtually addressed cache*, and it uses tags that are virtual addresses. However, a problem called *aliasing* may occur. Explain what *aliasing* is and why. (c) In today's computer systems, virtual memory and cache work together as a hierarchy. When the operating system decides to move a page back to disk, the contents of that page may have been brought into the cache already. What should the OS do with the contents that are in the cache?

(5) (20%) The following three instructions are executed using MIPS 5-stage pipeline.

1. ***lw*** ***$2,*** ***20($1)***
2. ***sub*** ***$4,*** ***$2,*** ***$5***
3. ***or*** ***$4,*** ***$2,*** ***$6***

Since there is one cycle delay between ***lw*** and ***sub***, a hazard detection unit is required. Furthermore, by the time the hazard is detected, ***sub*** and ***or*** may have already been fetched into the pipeline. Therefore it is also required to turn ***sub*** into a ***nop*** and delay the execution of ***sub*** and ***or*** by one cycle as shown below.

1. ***lw*** ***$2,*** ***20($1)***
2. ***nop***
3. ***sub*** ***$4,*** ***$2,*** ***$5***
4. ***or*** ***$4,*** ***$2,*** ***$6***

(a) In which stage should the hazard detection unit be placed? Why? (b) How can you turn ***sub*** into a ***nop*** in MIPS 5-stage pipeline? (c) How can you prevent ***sub*** and ***or*** from making progress and force these two instructions to repeat in the next clock cycle? (d) Explain why there is one cycle delay between ***lw*** and ***sub.***

(6) (20%) Answer the following questions briefly.

(a) Will addition "0010 + 1110" cause an overflow using the 4-bit two's complement

signed-integer format? (Simply answer yes or no). (5%)

(b) What would you get after performing *arithmetic right shift* by one bit on 1100_{two}? (5%)

(c) If one wishes to increase the *accuracy* of the floating-point numbers that can be represented, then he/she should increase the size of which part in the floating-point format? (5%)

(d) Name one event other than branches or jumps that the normal flow of instruction execution will be changed, e.g., by switching to a routine in the operating system. (5%)

(7) (15%) A MIPS instruction takes fives stages in a pipelined CPU design: (1) *IF*: instruction fetch, (2) *ID*: instruction decode/register fetch, (3) *ALU*: execution or calculate a memory address, (4) *MEM*: access an operand in data memory, and (5) *WB*: write a result back into the register. Label one appropriate stage in which each of the following actions needs to be executed. (Note that *A* and *B* are two source operands, while *ALUOut* is the output register of the *ALU*, *PC* is the program counter, *IR* is the instruction register, *MDR* is the memory data register, *Memory*[*k*] is the *k*-th word in the memory, and *Reg*[*k*] is the *k*-th registers in the register file.)

(a) *Reg*[*IR*[15-11]] = *MDR*; (5%)

(b) *ALUOut* = *PC* + (sign-extend (*IR*[15-0]) << 2); (5%)

(c) *Memory*[*ALUOut*] = *B*; (5%)