Problem 2.24 (20). Suppose the program counter (PC) is set to 0x2000 0000. Is it possible to use the jump (j) MIPS assembly instruction to set the PC to the address as 0x4000 0000? Is it possible to use the branch-on-equal (beq) MIPS assembly instruction to set the PC to this same address?

Answer: jump: no, beq: no

MIPS jump and branch instructions range

Problem 2.25 (20). The following instruction is not included in the MIPS instruction set:

\[ \text{rpt } \texttt{t2, loop # if(R[rs]>0) R[rs]=R[rs]-1, PC=PC+4+BranchAddr} \]

Note that the instruction should be interpreted as below:

If (R[rs]>0) {
    R[rs] = R[rs] - 1,
    PC = PC + 4 + BranchAddr
}

Problem 2.25.1 (10). If this instruction were to be implemented in the MIPS instruction set, what is the most appropriate instruction format?

Answer: i-type

Problem 2.25.2 (10). What is the shortest sequence of MIPS instructions that performs the same operation?

Answer:

```
addi $t2, $t2, -1
beq $t2, $0, loop
```

Problem 2.27 (20). Translate the following C code to MIPS assembly code. Use a minimum number of instructions. Assume that the values of a, b, i, and j are in registers $s0, $s1, $t0, and $t1, respectively. Also, assume that register $s2 holds the base address of the array D.

```c
1
```
for(i=0; i<a; i++)
    for(j=0; j<b; j++)
        D[4*j] = i + j;

Answer:

# a:$s0, b:$s1, D:$s2 i:$t0, j:$t1

 addi $t0, $0, 0
 beq $0, $0, TEST1
LOOP1: addi $t1, $0, 0
         beq $0, $0, TEST2
LOOP2: add $t3, $t0, $t1
         sll $t2, $t1, 4
         add $t2, $t2, $s2
         sw $t3, ($t2)
         add $t1, $t1, 1
TEST2: slt $t2, $t1, $s1
        bne $t2, $0, LOOP2
        addi $t0, $t0, 1
TEST1: slt $t2, $t0, $s0
        bne $t2, $0, LOOP1

Problem 2.28 (20). How many MIPS instructions does it take to implement the C code from Exercise 2.27? If the variables a and b are initialized to 10 and 1 and all elements of D are initially 0, what is the total number of MIPS instructions that is executed to complete the loop?

Answer: 14 instructions to implement and 158 instructions executed

Problem 2.29 (20). Translate the following loop into C. Assume that the C-level integer i is held in register $t1, $s2 holds the C-level integer called result, and $s0 holds the base address of the integer MemArray.

 addi $t1, $0, $0
LOOP: lw $s1, 0($s0)
       add $s2, $s2, $s1
       addi $s0, $s0, 4
       addi $t1, $t1, 1
       slti $t2, $t1, 100
       bne $t2, $s0, LOOP

Answer:
$s0$ holds the base address of MemArray, and $addi$ $s0$, $s0$, 4 instruction shift 4 byte to the right of the array. This means $s0 will point to next integer in the array. For example, currently $s0 points to MemArray[i]. After executing this instruction, $s0 will point to MemArray[i+1]
// assuming addr is held in register $s0
for (i=0; i<100; i++) {
    result = result + *addr;
    addr = addr + 1;
}

or

for (i=0; i<100; i++) {
    result = result + MemArray[i];
}