CSM151B HW2

Michael Zhou
UID: 804 663 317

2.24, 2.26.1, 2.26.3, and 2.46.1

2.24

The binary equivalent of 0x2000 0000 is
0010 0000 0000 0000 0000 0000 0000 0000

and the binary equivalent of 0x4000 0000 is
0100 0000 0000 0000 0000 0000 0000 0000

The J Jump instruction consists of 6 bits of opcode and 26 bits of offset

When a J type instruction is executed, the full 32-bit jump target address is formed by concatenating the higher order four bits of the program counter, the 26 bits of offset and two zeros at the lowest two bits.

Therefore **it is not possible to use the jump (j) MIPS assembly instruction to set the PC to address of 0x4000 0000.** Although jump can locate to full 32 bit address, it cannot jump to an address beyond 28 bits.

The branch-on-equal (beq) instruction consists of 6 bits of opcode, 5 bits of rs, 5 bits of rt, and 16 bits of offset.

However, the program cannot set the PC address from 0x2000 0000 to 0x4000 0000 within one beq instruction. Since the offset for beq is only 16 bits, the target address is smaller than the address can be branched to.Branch is an I-type instruction, which can only use 18 bits address.

2.26.1

```
LOOP:       slt  $t2, $0,  $t1
            beq  $t2, $0,  DONE
            subi $t1, $t1, 1
            addi $s2, $s2, 2
            j    LOOP
DONE:
```

In this case, $t2 is set to 1 if the value in $t1 is larger than zero
If $t2 becomes zero, break the loop and branch to DONE
Every once the condition is satisfied, subtract the value in $t1 by 1 and add $s2 by 2.
The loop goes through 10 times and $s2 is also added for 10 times in total. So the final value of $s2 is 20.

2.26.3
Assume that N is a positive number.
The number of instructions in the loop is 5 and the loop iterates for N times.
After N loops, slt and beq will operate once and jump to done.
Thus, the total number of instructions operated is 5N + 2.

2.46.1

CPI for different instructions:
Arithmetic: 1
Load/Store: 10
Branch: 3

Instruction breakdown:
Arithmetic: 500 million
Load/Store: 300 million
Branch: 100 million

ET = IC*CPI*CT
The original CPU time is calculated as:
ET = (500*1+300*10+100*3)*Orig Cycle Time

Because the number of arithmetic instructions are reduced by 25% and the cost of increasing cycle time by 10%, the new CPU time is adjusted into:
New ET = (500*(1-25%)+300*10+100*3)*(1+10%)Orig Cycle Time

Simplify, we get,
New ET = 4042.5*Orig Cycle Time

We conclude that after the adjustments of arithmetic instruction count & Cycle Time, the new CPU time is 4042.5 times slower than the original CPU time under this instruction breakdown. Therefore, it is not a good design choice.