CS420/520 Homework Assignment 2: ISAs

3.11 [10] <3.5> Consider the following fragment of C code:

for (i=0; i < = 100; i = i + 1) { a[i] = b[i] + c;}

Assume that *a* and *b* are arrays of words and the base address of *a* is in $a0 and the base address of *b* is in $a1. Register $t0 is associated with variable *i* and register $s0 with variable *c*. Write the code for MIPS. How many instructions are executed during the running of this code? How many memory data references will be made during execution?

S.1 [5] Show the three MIPS instruction formats, and explain the differences between three addressing modes of I-format instructions.

S.2 [5] What are key advantages of using registers in instructions?

S.3 [5] VAX instruction sets have an addressing mode, called *Displacement*. It is represented as

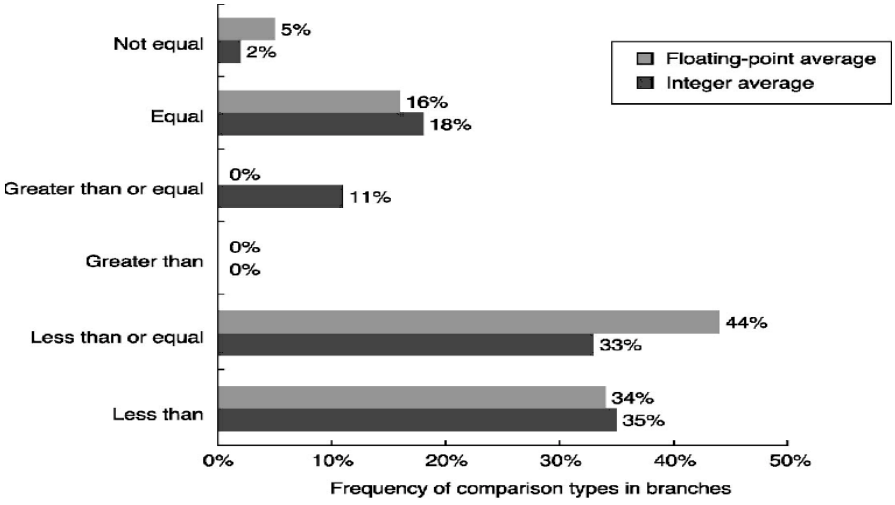
Add R4,100(R1) // function: R4 = R4+Mem[100+R1]

Please use MIPS instructions to fulfill the same function.

S.4 [5] How big-endian addressing and little-endian addressing differ? Give a representative machine for each addressing mode.

S.5 [5] Memory aside, where can the operands of instructions be from?

S.6 [5] What does the following diagram imply from an instruction set designer' perspective?



S. 7[5] What are typical elements of a machine instruction?

S.8 [5] What types of locations can hold source and destination operands?

S.9 [5] What general roles are performed by processor registers?

S.10 [20] For the following assume that values A, B, C, D, and E reside in memory. Also assume that instruction operation codes are represented in 8 bits, memory addresses are 64 bits, and register addresses are 6 bits.

[a] For each instruction set architecture shown in the following figure, how many addresses, or names, appear in each instruction for the code to computer C = A + B, and what is the total code size?

° Code sequence for C = A + B for four classes of instruction sets:

Stack	Accumulator	Register (register-memory)	Register (load-store)
Push A	Load A	Load R1,A	Load R1,A
Push B	Add B	Add R1,B	Load R2,B
Add	Store C	Store C, R1	Add R3,R1,R2
Pop C			Store C,R3

[b] Some of the instruction set architectures in the figure above destroy operands in the course of computation. This loss of data values from processor internal storage has performance consequences. For each architecture in the figure above, write the code sequence to computer C = A + B followed by D = A – E. In your code, mark each operand that is destroyed during execution and mark each "overhead" instruction that is included just to overcome this loss of data from processor internal storage. What is the total code size, the number of bytes of instructions and data moved to or from memory, the number of overhead instructions, and the number of overhead data bytes for each of your code sequences?

> ***Example solution: the following is the solution for the Stack architecture. Please show the solutions for other 3 architectures.***

b. Assume that each variable of A, B, C, D, and E is of size *n* bytes. Figures S.6

Stack Code	Comment	Code size (bits)	Data size (bytes)
Push A	load from memory	72	n
Push B		72	n
Add	operands A and B destroyed	8	0
Pop C	save result to memory	72	n
Push E		72	n
Push A	overhead instruction; reloading A onto stack	72	n
Sub	operands A and E destroyed; order of operands on stack determines which is minuend and subtrahend	8	0
Pop D		72	n

Figure S.6 Stack architecture code. The total code size is 448 bits (56 bytes), and the total data moved to or from memory is 6*n* bytes. There is 1 overhead instruction and 1 overhead data operand movement.

S. 11 [5] Several researchers have suggested that adding a register-memory addressing mode to a load-store machine might be useful. The idea is to replace sequences of

```
LOAD    R1, 0 (Rb)
ADD     R2, R2, R1      ;// given same rs and rd.
```

By:

```
ADD     R2, 0(Rb)       ;// given same rs and rd.
```

Please show a situation in a multiple instruction sequence where a load of R1 followed immediately by a use of R1 (with ADD/SUB operation) could NOT be replaced by a single instruction of the form proposed, assuming that the same opcode exists.

S.12 [5] Raise the basic advantages and disadvantages of the three conditional branch options, i.e., condition code (CC), condition register, and compare and branch.