# CSE301 – Computer Organization

## Tutorial 3

---

### Instruction Set Architecture (ISA)

#### MIPS Instruction Format Modification

If the instruction set of the MIPS processor is modified, the instruction format must also be changed. For each of the suggested changes, determine the size of the bit fields of a format instruction and find the total number of bits needed for each instruction.

**R-Type Instruction Format Modifications**

a. 8 Registers
b. 10-bit Immediate Constants
c. 128 Registers
d. All arithmetic instructions can use base addressing mode with the last argument (Example: add $a0, $a2, 0[$t1])

**I-Type Instruction Format Modifications**

a. 8 Registers
b. 10-bit Immediate Constants
c. 128 Registers
d. All arithmetic instructions can use base addressing mode with the last argument (Example: add $a0, $a2, 0[$t1])

**Why could the suggested change decrease the size of a MIPS assembly program? Why could the suggested change increase the size of a MIPS assembly program?**

---

## Part 2: Questions with Answers

### Instruction Set Architecture (ISA)

#### MIPS Instruction Format Modification

If the instruction set of the MIPS processor is modified, the instruction format must also be changed. For each of the suggested changes, determine the size of the bit fields of a format instruction and find the total number of bits needed for each instruction.

**R-Type Instruction Format Modifications**

a. 8 Registers

No. of bits for addressing registers = 3bits
Total bits for instruction = 6 + 3 + 3 + 3 + 5 + 6 = 26bit


b. 10-bit Immediate Constants

**No modification**,, R-type instructions don't have immediate constant field

c. 128 Registers

No. of bits for addressing registers = 7bits
Total bits for instruction = 6 + 7 + 7 + 7 + 5 + 6 = 38bit


d. Base Addressing Mode Support

Add a field for immediate value 16bit
Total bits for instruction = 6 + 5 + 5 + 5 + 5 + 6 + 16 = 48bit


R-type instructions use the opcode 0 and the funct to denote the operation. Using an unused opcode (there are many) and the same funct combinations, all instructions with base addressing can be added. The length of the immediate was unspecified -- 16 is a good choice since it is a half-word, and the size used in I-type instructions.

However, no instruction that uses the shamt field can have base addressing as asked in the question, so it is possible to use the shamt field and allow only 5-bit offsets and retain the instruction length.

**I-Type Instruction Format Modifications**

a. 8 Registers

No. of bits for addressing registers = 3bits
Total bits for instruction = 6 + 3 + 3 + 16 = 28bit


b. 10-bit Immediate Constants

Total bits for instruction = 6 + 5 + 5 + 10 = 26bit


c. 128 Registers

No. of bits for addressing registers = 7bits
Total bits for instruction = 6 + 7 + 7 + 16 = 36bit


d. Base Addressing Mode Support

No change using the assumption that the definition of an immediate must be an integer constant.
37 bits for an instruction of the form addi $a0, $s1, 0($t1). We will need to add another5 bits to the register data.


**Why could the suggested change decrease the size of a MIPS assembly program? Why could the suggested change increase the size of a MIPS assembly program?**

**8 Registers**

Smaller instruction encoding may mean smaller programs -- this may be difficult in practice since the reductions aren't byte-aligned, so addressing subsequent instructions is difficult. It also breaks the MIPS tenet of fixed-length instructions. Programs may increase in size because fewer registers means more data will 'spill' into memory, leading to more load/store instructions.

**10 bit immediate constants**

Smaller instruction encoding may mean smaller programs -- this may be difficult in practice since the reductions aren't byte-aligned, so addressing subsequent instructions is difficult. It also breaks the MIPS tenet of fixed-length instructions. Programs may increase in size because to use immediate numbers larger than 10-bits, you need additional shift/or instructions.

**128 Registers**

Longer instruction-encoding means longer programs that do the same operations. The instruction formats may be even larger than indicated above since the increases aren't byte-aligned and addressing subsequent instructions is difficult. It also breaks the MIPS tenet of fixed-length instructions. Programs may decrease in size because more registers means you can 'hold' onto more data in the processor without going out to memory, leading to fewer load/store instructions.

**Base Addressing Mode Support**

A larger instruction format means programs that cannot make use of the new addressing mode will be longer. This may be further exacerbated by extending I and J-type instructions to 48-bits as well, to keep instructions fixed-length. However, based-addressing allows many load operations to be combined into an arithmetic instruction -- this is a reduction from 2*32 bytes to 48 bytes for each such reduction. A program that heavily uses this might grow smaller. If I and J-types are made 48-bytes as well, 32-bit immediate constants and longer jump offsets may also contribute to reducing the program length (though unlikely).

If the shamt field is used, the program will not grow larger with this addition, though it may grow smaller. However, adding base-addressing still adds complexity to the processor that may slow the program down. It is also less useful with just 5-bit offsets.