**RISC Philosophy**

Lotsa Registers

All Instructions same size (32 bits)

Load (and any other instruction take parameters from right to left) / store (from left to right)

Pseudo instruction:

Move $t1, $t0 ( t1 = t0 )

Load

la $t4, joe – get the address of joe and store in to $t4

lw $t0, 0($t4) or lw $t0, $t4,0 (same) - dereference and store the value of the address in $t4 into $t0

J instruction 26 bits address –actually 28 bits means 256 mega bytes limit

**Data Types**

Zero flags – not in MIPS

Carry flags – not in MIPS (only to signed number) (-2^3 2^2 2^1 2^0)

Overflow – add two number of the same sign but result in a different sign

The address got 26 bit address, in fact got 28 bit because offset\*4 – range: 16MBs

Jalr – Jump and link – jump and save the pc to register 31

Branch and jump is signed extended

Branch only have equal and not equal for hardware

Blt – pseudo instruction

**Test Instructions**

Slt $t0, $t1, $t2 (if t1 < t2 : t0 = 1 if not t0 = 0)

Slti $t0, $t1, 0x50

**Memory**

Immediate instruction

Lw $t0, 0($t4) (the address in t4 and add to te offset “0” and put the target value in t0)

Sw

**Psuedo**

“bare-metal”

$at is the assembler register reserved for pseudo instructions.

**Shift**

Srlv $t1, $t2, $t3

Rd rt rs (#number of time shifts)

**Add to 32 bits -> 64 bits result**

**A a0 a1**

**B a2 a3**

Add $v1, $a1, $a3

If v1 less than either the 2 argument passed in a1 or a2 -> carry happen

Sltu $v0, $v1, $a1 (store 1 into v0 if v1 less than argument)

Add the carry with each upper value to get the final result

Addu $v0, $v0, $a2 (add carry with the first high arg)

Addu $v0, $a0, $v0 (add the previous ans to another high arg ->final result)

**Stack**