Computer Organization (IS F242)

Lect 24: MIPS Architecture

R – Type Instruction

add rd, rs, rt	(0, rs, rt, rd, 0, 32) (with overflow)
addu rd, rs, rt	(0, rs, rt, rd, 0, 33) (without OF)
and rd, rs, rt	(0, rs, rt, rd, 0, 36)
clo rd, rs	(28, rs, 0, rd, 0, 33)
Count leading ones	
clz rd, rs	(28, rs, 0, rd, 0, 32)
Count leading zeros	
div rs, rt	(0, rs, rt, 0, 0, 26) (with overflow)
divu rs, rt	(0, rs, rt, 0, 0, 27) (without OF)
mult rs, rt	(0, rs, rt, 0, 0, 24)
multu rs, rt	(0, rs, rt, 0, 0, 25) (Unsigned Mult)
mul rd, rs, rt	(28, rs, rt, rd, 0, 2) (without OF)

madd rs, rt	(28, rs, rt, 0, 0, 0)
maddu rs, rt	(28, rs, rt, 0, 0, 1) (Unsigned)

  - Multiply Add
  - Multiply rs and rt and add the resulting 64 bit product to the 64 bit value in the concatenated registers lo and hi

msub rs, rt	(28, rs, rt, 0, 0, 4)
msubu rs, rt	(28, rs, rt, 0, 0, 5)

  - Multiply Subtract
  - Multiply rs and rt and subtract the resulting 64 bit product from the 64 bit value in the concatenated registers lo and hi

nor rd, rs, rt	(0, rs, rt, rd, 0, 39)
or rd, rs, rt	(0, rs, rt, rd, 0, 37)
sll rd, rt, shamt	(0, rs, rt, rd, shamt, 0)

  - Shift left logical – does not preserve the number’s sign bit

sllv rd, rt, rs	(0, rs, rt, rd, 0, 4)

  - Shift left logical variable

- sra rd, rt, shamt (0, rs, rt, rd, shamt, 3)
  - Shift right arithmetic – Empty position in MSB is filled with a copy of the original MSB
- srav rd, rt, rs (0, rs, rt, rd, 0, 7)
  - Shift right arithmetic variable
- srl rd, rt, shamt (0, rs, rt, rd, shamt, 2)
  - Shift right logical - does not preserve the number's sign bit
- srlv rd, rt, rs (0, rs, rt, rd, 0, 4)
  - Shift right logical variable
- sub rd, rs, rt (0, rs, rt, rd, 0, 34) (with overflow)
- subu rd, rs, rt (0, rs, rt, rd, 0, 35) (with out OF)
- xor rd, rs, rt (0, rs, rt, rd, 0, 38)
- slt rd, rs, rt (0, rs, rt, rd, 0, 42)
- sltu rd, rs, rt (0, rs, rt, rd, 0, 43) (Unsigned)
  - Set less than

- mfhi rd (0,0,0,rd,0,16)
- mflo rd (0,0,0,rd,0,18)
- mthi rs (0,rs,0,0,0,17)
- mtlo rs (0,rs,0,0,0,19)
- movn rd, rs, rt (0,rs,rt,rd,0,11)
  - Move conditional not zero
  - move rs to rd if rt is not 0
- movz rd, rs, rt (0,rs,rt,rd,0,10)
  - Move conditional zero
  - move rs to rd if rt is 0

- addi rt, rs, imm (8, rs, rt, imm) (with overflow)
- addiu rt, rs, imm (9, rs, rt, imm) (without overflow)
- andi rt, rs, imm (12, rs, rt, imm)
- ori rt, rs, imm (13, rs, rt, imm)
- xori rt, rs, imm (14, rs, rt, imm)
- lui rt, imm (15, 0, rt, imm)
  - Load upper immediate. Loads the lower half word of the imm to the upper half word of rt. Lower bits of rt are set to 0
- slti rt, rs, imm (10, rs, rt, imm)
- sltiu rt, rs, imm (11, rs, rt, imm) (Unsigned)
  - Set less than immediate
- beq rs, rt, label (4, rs, rt, offset)
  - Branch the number of instructions specified by the offset if rs is equal to rt
- bgeq rs, label (1, rs, 1, offset)
  - Branch the number of instructions specified by the offset if rs is greater than or equal to zero

- bgezal rs, label (1, rs, 17, offset)
  - Branch on greater than equal zero and link. Save the address of the next instruction in register 31
- bgtz rs, label (7, rs, 0, offset)
  - Branch on greater than zero
- blez rs, label (6, rs, 0, offset)
  - Branch on less than equal zero
- bltzal rs, label (1, rs, 16, offset)
  - Branch on less than zero and link. Save the address of the next instruction in register 31
- bltz rs, label (1, rs, 0, offset)
  - Branch on less than zero
- bne rs, rt, label (5, rs, rt, offset)
  - Branch on not equal

lb rt, address	(32, rs, rt, offset)
Load byte	
lbu rt, address	(36, rs, rt, offset)
Load unsigned byte	
lh rt, address	(33, rs, rt, offset)
Load half word	
lhu rt, address	(37, rs, rt, offset)
Load unsigned half word	
lw rt, address	(35, rs, rt, offset)
Load word	
sb rt, address	(40, rs, rt, offset)
Store byte	
sh rt, address	(41, rs, rt, offset)
Store half word	
sw rt, address	(43, rs, rt, offset)
Load word	

Jump

- j imm # Jump absolute
- jal imm # Jump and link ($ra ← PC)
- jr rs # Jump register (PC ← rs)
- jalr rs, rt # Jump register and link (rt ← PC, PC ← rs)
- All jumps are absolute
  - 26 bits absolute address
  - 32 bits??
- All branches are relative to PC
  - 16 bit signed offset