ICS 233
Computer Architecture & Assembly Language

MIPS PROCESSOR INSTRUCTION SET

ICS 233
Computer Architecture & Assembly Language

Lecture 8

Lecture Outline

- ❑ **MIPS Compare or Set Less than instructions**
- ❑ **MIPS Conditional Branch Instructions**
- ❑ **MIPS Instruction J-Type Format**
- ❑ **MIPS Unconditional Jump Instructions**

Compare : Set on Less Than Instructions

- **MIPS also provides set on less than instructions**

`slt rd,rs,rt`	if (rs < rt) rd = 1 else rd = 0
`sltu rd,rs,rt`	unsigned <
`slti rt,rs,im16`	if (rs < im^{16}) rt = 1 else rt = 0
`sltiu rt,rs,im16`	unsigned <

- **Signed / Unsigned Comparisons can produce different results**

Assume `$s0 = 1` and `$s1 = -1 = 0xffffffff`

`slt $t0,$s0,$s1`	results in	`$t0 = 0`
`stlu $t0,$s0,$s1`	results in	`$t0 = 1`

MIPS – Comparison Instructions

- **slt** (set less than)
  - **Instruction Mnemonic :**
    slt rd, rs, rt ;where rs, rt, rd are registers,
  - **Meaning :**
    if (rs < rt) then rd = 1 else rd = 0
  - **Example :**
    slt $s1, $s2, $s3 ; if ($s2 < $s3) then $s1=1 else $s1=0

- **slti** (set less than immediate)
  - **Instruction Mnemonic :**
    slti rd, rs, const ;where rs, rd are registers,
    ; const is a 16-bit constant
  - **Meaning :**
    if (rs < const) then rd = 1 else rd = 0
  - **Example :**
    slti $s1, $s2, 100 ; if ($s2 < 100) then $s1=1 else $s1=0

MIPS – Comparison Instructions

- **sltu** (set less than unsigned)
  - **Instruction Mnemonic :**
    sltu rd, rs, rt ;where rs, rt, rd are registers,
  - **Meaning :**
    if (rs < rt) then rd = 1 else rd = 0
  - **Example :**
    sltu $s1, $s2, $s3 ; if ($s2 < $s3) then $s1=1 else $s1=0

- **sltiu** (set less than immediate unsigned)
  - **Instruction Mnemonic :**
    sltiu rd, rs, const ;where rs, rd are registers,
    ; const is a 16-bit constant
  - **Meaning :**
    if (rs < const) then rd = 1 else rd = 0
  - **Example :**
    sltiu $s1, $s2, 100 ; if ($s2 < 100) then $s1=1 else $s1=0

SLT Instructions

Instruction	Meaning	Format					
slt rd, rs, rt	rd=(rs<rt?1:0)	$op^6 = 0$	rs^5	rt^5	rd^5	0	0x2a
sltu rd, rs, rt	rd=(rs<rt?1:0)	$op^6 = 0$	rs^5	rt^5	rd^5	0	0x2b
slti rt, rs, imm^{16}	rt=(rs<imm?1:0)	0xa	rs^5	rt^5	imm^{16}		
sltiu rt, rs, imm^{16}	rt=(rs<imm?1:0)	0xb	rs^5	rt^5	imm^{16}		

Conditional Branch Instructions

- **MIPS compare and branch instructions:**

  `beq Rs,Rt,label` branch to `label` if (`Rs == Rt`)

  `bne Rs,Rt,label` branch to `label` if (`Rs != Rt`)

- **MIPS compare to zero & branch instructions**

  Compare to zero is used frequently and implemented efficiently

  `bltz Rs,label` branch to `label` if (`Rs < 0`)

  `bgtz Rs,label` branch to `label` if (`Rs > 0`)

  `blez Rs,label` branch to `label` if (`Rs <= 0`)

  `bgez Rs,label` branch to `label` if (`Rs >= 0`)

- No need for **beqz** and **bnez** instructions. Why?

MIPS – Conditional branch Instructions

- **beq** (branch on equal)
- **Instruction Mnemonic :**

  beq rd, rs, addr ;where rs, rd are registers,
  ; addr is the label of the target location
- **Meaning :**

  if (rd == rs) then branch to label
  i.e., goto PC + 4 + const*4 (i.e., PC = Updated PC + offset)
- **Example :**

  beq $s1, $s2, up ; if ($s1 = $s2) goto target location up

- **bne** (branch on not equal)
- **Instruction Mnemonic :**

  bne rd, rs, addr ;where rs, rd are registers,
  ;addr is the label of the target location
- **Meaning :**

  if (rd != rs) then branch to label
  i.e., goto PC + 4 + const*4 (i.e., PC = Updated PC + offset)
- **Example :**

  bne $s1, $s2, loop ; if ($s1 != $s2) goto target location loop

MIPS – Conditional branch Instructions

- **bgtz** (branch on greater than zero)
- **Instruction Mnemonic :**

  bgtz rd, addr ;where rd is a register,
  ; addr is the label of the target location
- **Meaning :**

  if (rd > 0) then branch to label
  i.e., goto PC + 4 + const*4 (i.e., PC = Updated PC + offset)
- **Example :**

  bgtz $s1, up ; if ($s1 > 0) goto target location up

- **bgez** (branch on greater than or equal to zero)
- **Instruction Mnemonic :**

  bgez rd, addr ;where rd is a register,
  ; addr is the label of the target location
- **Meaning :**

  if (rd >= 0) then branch to label
  i.e., goto PC + 4 + const*4 (i.e., PC = Updated PC + offset)
- **Example :**

  bgez $s1, loop ; if ($s1 >= 0) goto target location loop

MIPS – Conditional branch Instructions

- **bltz** (branch on less than zero)
  - **Instruction Mnemonic :**

    bltz rd, addr ;where rd is a register,
    ; addr is the label of the target location
  - **Meaning :**

    if (rd < 0) branch to label
    i.e., goto PC + 4 + const*4 (i.e., PC = Updated PC + offset)
  - **Example :**

    bltz $s1, up ; if ($s1 < 0) goto target location up

- **blez** (branch on less than or equal to zero)
  - **Instruction Mnemonic :**

    blez rd, addr ; where rd is a register,
    ; addr is the label of the target location
  - **Meaning :**

    if (rd <= 0) branch to label
    i.e., goto PC + 4 + const*4 (i.e., PC = Updated PC + offset)
  - **Example :**

    blez $s1, loop ; if ($s1 <= 0) goto target location loop

Branch Instructions

Instruction	Meaning	Format			
beq rs, rt, label	branch if (rs == rt)	$op^6 = 4$	rs^5	rt^5	imm^{16}
bne rs, rt, label	branch if (rs != rt)	$op^6 = 5$	rs^5	rt^5	imm^{16}
blez rs, label	branch if (rs<=0)	$op^6 = 6$	rs^5	0	imm^{16}
bgtz rs, label	branch if (rs > 0)	$op^6 = 7$	rs^5	0	imm^{16}
bltz rs, label	branch if (rs < 0)	$op^6 = 1$	rs^5	0	imm^{16}
bgez rs, label	branch if (rs>=0)	$op^6 = 1$	rs^5	1	imm^{16}

More on Branch Instructions

- **MIPS hardware does NOT provide instructions for …**

`blt, bltu`	branch if less than	(signed/unsigned)
`ble, bleu`	branch if less or equal	(signed/unsigned)
`bgt, bgtu`	branch if greater than	(signed/unsigned)
`bge, bgeu`	branch if greater or equal	(signed/unsigned)

Can be achieved with a sequence of 2 instructions

- How to implement:
- Solution:

```
blt $s0,$s1,label
slt $at,$s0,$s1
bne $at,$zero,label
```

- How to implement:
- Solution:

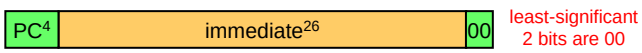
```
ble $s2,$s3,label
slt $at,$s3,$s2
beq $at,$zero,label
```


J-Type Format

- **J-type format is used for unconditional jump instruction**:

```
j    label      # jump to label
  . . .
label:
```

- **26-bit immediate value is specified in the instruction**
  - Immediate constant specifies **address of target instruction**
- **Program Counter (PC) is modified as follows:**
  - Next PC =
  - Upper 4 most significant bits of PC are unchanged



MIPS – Unconditional Jump Instructions

❑ **j (jump to target address)**

➢ **Instruction Mnemonic :**

j addr **;where addr is the label of the target address**

➢ **Meaning :**

jump to the target address location specified by the label addr in the instruction

➢ **Example :**

j loop **; goto location having the label loop**

Jump and Branch Limits

- Jump Address Boundary = 2^{26} instructions = 256 MB
  - Text segment cannot exceed 2^{26} instructions or 256 MB
  - Upper 4 bits of PC are unchanged

Target Instruction Address

- Branch Address Boundary
  - Branch instructions use I-Type format (16-bit immediate constant)
  - PC-relative addressing:

    - Target instruction address = PC + 4×(1 + immediate^{16})
    - Count number of instructions to branch from next instruction
    - Positive constant => Forward Branch,
    - Negative constant => Backward branch
    - At most $\pm 2^{15}$ instructions to branch (most branches are near)