Assembly Language Lecture Series: RV32I: Control Transfer instructions

Roger Luis Uy
College of Computer Studies
De La Salle University
Manila, Philippines

'2112

Control Transfer instructions

Control transfer (unconditional branch) instructions				
JAL	JALR			

Conditional transfer (conditional branch) instructions				
BEQ	BNE	BLT	BLTU	BGE
BGEU				

Unconditional branch instructions

JAL *rd*, *offset*

- *offset* is 20-bit offset encodes signed offset in multiples of 2 bytes
- The offset is sign-extended and added to the address of the branch instruction to give the target address.
- Various pseudoinstructions are available for jump, call and ret

JAL instruction (view as call)

JAL *rd*, *offset*

- JAL stores the address of the instruction following the jump (pc+4) into register *rd*. The standard software calling convention uses x1 as the return address register and x5 as an alternate link register.
- The pseudo-instruction call L1 can be use as alternative to JAL x1, label.
- The pseudo-instruction ret [translated as JALR x0, 0(x1)] is use as a return.

```
Example:
addi x10, x0, 0x05
JAL x1, L1
NOP
NOP
:
:
L1: addi x11, x0, 0x07
      JALR x0, 0(X1)


After execution:
x10 = 00000005
x11 = 00000007
```

```
Example (the same):
addi x10, x0, 0x05
call L1
NOP
NOP
:
:
L1: addi x11, x0, 0x07
      ret


After execution:
x10 = 00000005
x11 = 00000007
```

JAL instruction (view as jump)

JAL *rd*, *offset*

- JAL stores the address of the instruction following the jump (pc+4) into register *rd*. The standard software calling convention uses x1 as the return address register and x5 as an alternate link register.
- The pseudo-instruction J *offset* (translated as JAL x0, offset) is use as unconditional jump.

Example:

addi x10, x0, 0x05

JAL x0, L1

NOP

addi x11, x0, 0x06

L1: addi x11, x0, 0x07

After execution:

x10 = 00000005

x11 = 00000007

Example (the same):

addi x10, x0, 0x05

J L1

NOP

addi x11, x0, 0x06

L1: addi x11, x0, 0x07

After execution:

x10 = 00000005

x11 = 00000007

JALR instruction

JALR *rd*, *offset(rs)*

- JALR (jump and link register): the target address is obtained by adding the sign-extended 12-bit immediate to the register *rs*, then setting the least-significant bit of the result to zero.
- The address of the instruction following the jump (pc+4) is written to register *rd*.
- Register x0 can be used as the destination if the result is not required.
- The pseudo-instruction ret [translated as JALR x0, 0(x1)] is use as a return.

Example:

```
addi x10, x0, 0x05
JAL x1, L1
NOP
NOP
:
:
L1: addi x11, x0, 0x07
      JALR x0, 0(x1)
```

After execution:

x10 = 00000005

x11 = 00000007

Example (the same):

```
addi x10, x0, 0x05
call L1
NOP
NOP
:
:
L1: addi x11, x0, 0x07
      ret
```

After execution:

x10 = 00000005

x11 = 00000007

Pseudoinstruction

Pseudo-instruction	Equivalent	description
j offset	jal x0, offset	Jump
jal offset	jal x1, offset	Jump and link
jr rs	jalr x0, 0(rs)	Jump register
jalr rs	jalr x1, 0(rs)	Jump and link register
ret	jalr x0, 0(x1)	Return from subroutine
call offset	Auipc x1, offset[31:12]+offset[11] Jalr x1, offset[11:0](x1)	Call far-away subroutine

Conditional branch instructions

- *Bxx rs1*, *rs2*, *offset*
- Branch instructions compare two registers. Branch taken if:
  - BEQ (*rs1* == *rs2*); BNE (*rs1* != *rs2*)
  - BLT (rs1 < rs2; signed); BLTU (rs1 < rs2; unsigned)
  - BGE (rs1 > rs2; signed); BGEU (rs1 > rs2; unsigned)
- *offset* is 12-bit offset encodes signed offset in multiples of 2 bytes
- The offset is sign-extended and added to the address of the branch instruction to give the target address.
- BGT, BGTU, BLE, BLEU are pseudoinstructions by reversing the operands to BLT, BLTU, BGE, BGEU
- BGT *rs1*, *rs2*, *offset* (translated as BLT *rs2*, *rs1*, *offset)*
- Other pseudoinstructions involving conditional branch with 0
  - BEQZ, BNEZ, BLEZ, BGEZ, BLTZ, BGTZ
  - example syntax: BEQZ *rs*, *offset* (translated as BEQ *rs*, x0, *offset*)

Conditional branch instruction

- *Bxx rs1, rs2, offset*
- BEQ, BNE, BLT, BLTU, BGE, BGEU
- The pseudo-instructions are available for other types of conditional branch

```
Example:
addi x10, x0, 0xFFFFFFFF
addi x11, x0, 0x05
BGE x10, x11, L1
NOP
addi x12, x0, 0x08
J L2
:
:
L1: addi x12, x0, 0x07
L2:     :
        :
After execution:
x10 = FFFFFFFF
x11 = 00000005
X12 = 00000008
```

```
Example:
addi x10, x0, 0xFFFFFFFF
addi x11, x0, 0x05
BGEU x10, x11, L1
NOP
addi x12, x0, 0x08
J L2
:
:
L1: addi x12, x0, 0x07
L2:     :
        :
After execution:
x10 = FFFFFFFF
x11 = 00000005
X12 = 00000007
```