RISC V Architecture

Mahesh Awati
Department of Electronics and Communication Engg.

RISC V ARCHITECTURE

UNIT 2 – Instructions: The Language of Computer

Mahesh Awati
Department of Electronics and Communication Engineering

Instructions – Language of Computer

Logical Operations

Logical operations	C operators	Java operators	RISC-V Instructions
Shift left	<<	<<	sll, slli
Shift right	>>	>>>	srl, srli
Shift right arithmetic	>>	>>	sra, srai
Bit-by-bit AND	&	&	and, andi
Bit-by-bit OR	|	|	or, ori
Bit-by-bit XOR	^	^	xor, xori
Bit-by-bit NOT	~	~	xori

Instructions – Language of Computer

Logical Operations

Logical AND,OR,XOR and NOT – R and I type

Syntax	and rd, rs1,rs2
Operation:	AND is a bit-by-bit operation that leaves a 1 in the result only if both bits of the operands are 1. In this case, the content of source register rs2 is ANDed with rs1 logically and the result is stored in destination register rd
Example:	and x9, x10,x11; x9=x10 & x11
Before Execution	
x10= =	00000000 00000000 00001101 11000000b
X11 =	00000000 00000000 00111100 00000000b
x9 =	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
After Execution	
x10= =	00000000 00000000 00001101 11000000b
X11 =	00000000 00000000 00111100 00000000b
x9 =	00000000 00000000 00001100 00000000b

Use: AND can apply a bit pattern to a set of bits to force 0s where there is a 0 in the bit pattern.
This called as Masking

Instructions – Language of Computer

Logical Operations

Logical AND,OR,XOR and NOT

Syntax	or rd, rs1,rs2
Operation:	OR is a bit-by-bit operation that places a 1 in the result if *either* operand bit is a 1. In this case, the content of source register rs2 is ORed with rs1 logically and the result is stored in destination register rd
Example:	or x9, x10,x11; x9= x10 | x11
Before Execution	
x10= =	00000000 00000000 00001101 11000000b
X11 =	00000000 00000000 00111100 00000000b
x9 =	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
After Execution	
x10= =	00000000 00000000 0000**1101 11**000000b
X11 =	00000000 00000000 00**1111**00 00000000b
x9 =	00000000 00000000 00111101 11000000b

Use: Setting bits

Instructions – Language of Computer

Logical Operations

Logical AND,OR,XOR and NOT

Syntax	xor rd, rs1,rs2
Operation:	A logical bit-by bit operation with two operands that calculates the exclusive OR of the two operands. That is, it calculates a 1 only if the values are different in the two operands. In this case, the content of rs2 is XORed with rs1 logically and the result is stored in destination register rd
Example:	xor x9, x10,x11; x9= x10 ^ x11
Before Execution	
x10= =	00000000 00000000 00001101 11000000b
X11 =	00000000 00000000 00111100 00000000b
x9 =	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
After Execution	
x10= =	00000000 00000000 00001101 11000000b
X11 =	00000000 00000000 00111100 00000000b
x9 =	00000000 00000000 00110001 11000000b

Logical AND,OR,XOR and NOT

RISC-V also provides the instructions *and immediate* (andi), *or immediate* (ori), and *exclusive or immediate* (xori).

- AND x9, x10,imm_{12}
- OR x9, x10, imm_{12}
- XOR x9, x10, imm_{12}

Instructions – Language of Computer

Examples

For the following C statement, write the corresponding RISC-V assembly code. Assume that the C variables f, g, and h, have already been placed in registers x5, x6, and x7 respectively. Use a minimal number of RISC-V assembly instructions.
f = g + (h − 5);

```
addi x5, x7,-5
add x5, x5, x6
```

Variables	Registers
f	x5
g	x6
h	x7

Instructions – Language of Computer

Examples

For the following C statement, write the corresponding RISC-V assembly code. Assume that the variables f, g, h, i, and j are assigned to registers x5, x6, x7, x28, and x29, respectively. Assume that the base address of the arrays A and B are in registers x10 and x11, respectively.

B[8] = A[i–j];

```
sub x30, x28, x29 // compute i-j  (x28 – x29)
slli x30, x30, 2      // 32 bit data at [i-j]th element of Array A is at offset 4x[i-j]
add x3, x30, x10  // physical address of A[i-j] = Base Address + 4 x [i-j]
lw x30, 0(x3)         // load A[i-j]
sw x30, 32(x11)  // store in the content of x30 into B[8] =B[BA + 4x8] = B[x11 + 32]
```

Memory	0x0000	0x0004	0x0008	……	(i-j)x4	[(i-j)+1]x4	[(i-j)+2]
Element	0	1	2	……	(i-j)	[(i-j)+1]	[(i-j)+2]
A[]	A[0]	A[1]	A[2]	……	A[i-j]	A[(i-j)+1]	A[(i-j)+2]

Variables	Registers
f	x5
g	x6
h	x7
i	x28
j	x29
Base Address of A	x10
Base Address of B	x11

Instructions – Language of Computer

Examples

Assume that registers x5 and x6 hold the values x80000000 and 0xD0000000, respectively.

a) What is the value of x30 for the following assembly code?

```
add x30, x5, x6
```

b) Is the result in x30 the desired result, or has there been overflow?

c) For the contents of registers x5 and x6 as specified above, what is the value of x30 for the following assembly code?

```
sub x30, x5, x6
```

d) Is the result in x30 the desired result, or has there been overflow?

e) For the contents of registers x5 and x6 as specified above, what is the value of x30 for the following assembly code?

```
add x30, x5, x6
add x30, x30, x5
```

f) Is the result in x30 the desired result, or has there been overflow?

Variables	Registers
f	x5
g	x6
h	x7
i	x28
j	x29
Base Address of A	x10
Base Address of B	x11

a) 0x50000000
b) overflow
c) 0xB0000000
d) no overflow
e) 0xD0000000
f) overflow

THANK YOU

Mahesh Awati
Department of Electronics and Communication

mahesha@pes.edu

+91 9741172822