RISC V Architecture

Dr. Santhameena.S

Department of Electronics and Communication Engineering

RISC V Architecture

UNIT 3: Instructions: The Language of Computer

Dr. Santhameena.S
Department of Electronics and Communication Engineering

Instructions – Language of Computer

32 bit constants

- Most constants are small
  - 12-bit immediate is sufficient
- For the occasional 32-bit constant

  lui rd, constant
  - Copies 20-bit constant to bits [31:12] of rd
  - Extends bit 31 to bits [63:32]
  - Clears bits [11:0] of rd to 0

```
lui x19, 976  // 0x003D0
```

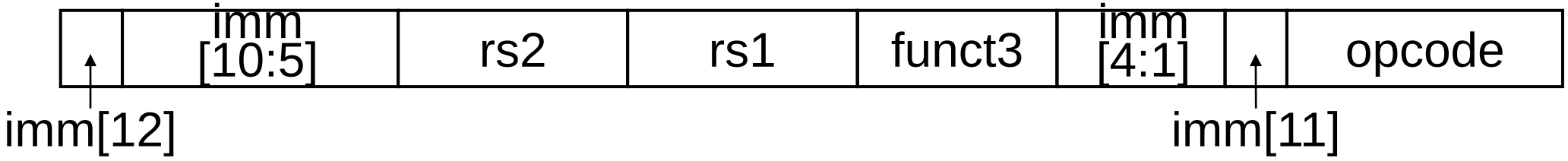
0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0011 1101 0000	0000 0000 0000

```
addi x19,x19,128  // 0x500
```

0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0011 1101 0000	0101 0000 0000

Instructions – Language of Computer
Branch Addressing

- Branch instructions specify
  - Opcode, two registers, target address
- Most branch targets are near branch
  - Forward or backward
- SB format:



- PC-relative addressing
  - Target address = PC + immediate × 2

Branch Addressing

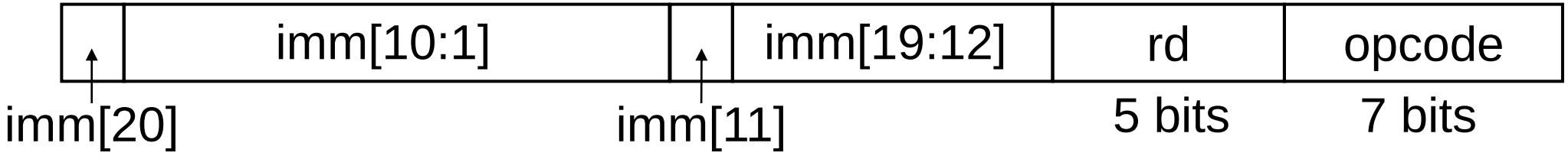
```
bne  x10, x11, 2000  // if x10 != x11, go to location 2000ten = 0111 1101 0000
```

0011111	01011	01010	001	01000	1100111
imm[12:6]	rs2	rs1	funct3	imm[5:1]	opcode

where the opcode for conditional branches is 1100111_{two} and `bne`'s funct3 code is 001_{two}.

Jump Addressing

- Jump and link (jal) target uses 20-bit immediate for larger range
- UJ format:



- For long jumps, eg, to 32-bit absolute address
  - lui: load address[31:12] to temp register
  - jalr: add address[11:0] and jump to target

Like the SB-type format, the UJ-type format's address operand uses an unusual immediate encoding, and it cannot encode odd addresses. So,

```
jal  x0, 2000  // go to location 2000ten = 0111 1101 0000
```

could be assembled into the U format (Section 4.4 will show the actual format for jal):

00000000001111101000	00000	1101111
imm[20:1]	rd	opcode

Instructions – Language of Computer
Showing Branch Offset in Machine Language

Here is a traditional loop in C:

```
while (save[i] == k)
      i += 1;
```

- It is compiled into RISC-V assembler code as,

```
Loop:slli x10, x22, 2     // Temp reg x10 = i * 4
     add  x10, x10, x25   // x10 = address of save[i]
     lw   x9, 0(x10)      // Temp reg x9 = save[i]
     bne  x9, x24, Exit   // go to Exit if save[i] != k
     addi x22, x22, 1     // i = i + 1
     beq  x0, x0, Loop    // go to Loop
Exit:
```

Instructions – Language of Computer
Showing Branch Offset in Machine Language

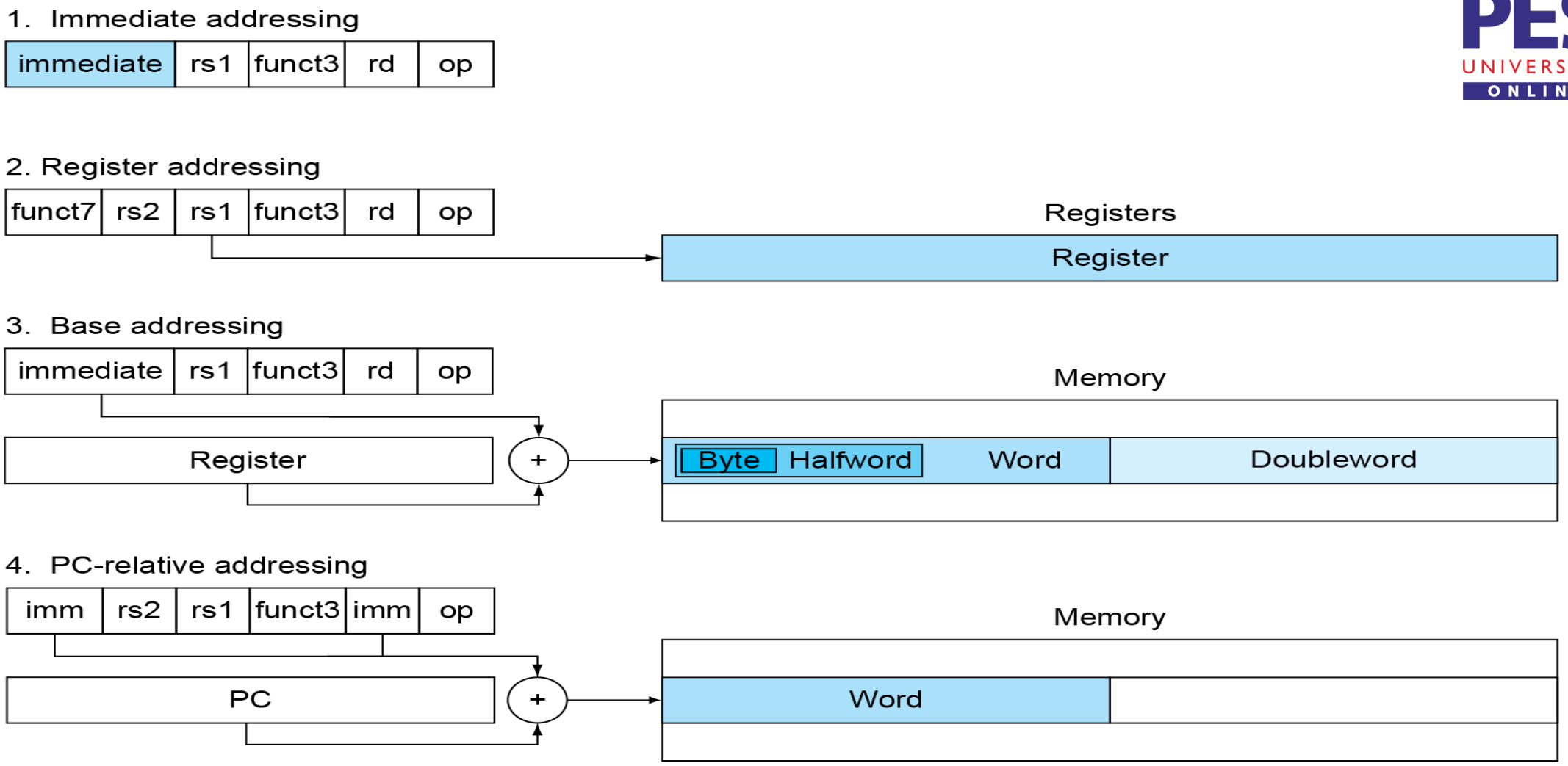
If we assume we place the loop starting at location 80000 in memory, what is the RISC-V machine code for this loop?

The assembled instructions and their addresses are:

```
Loop:slli x10, x22, 2
     add  x10, x10, x25
     lw   x9, 0(x10)
     bne  x9, x24, Exit
     addi x22, x22, 1
     beq  x0, x0, Loop
Exit:
```

Address	Instruction					
80000	0000000	00010	10110	001	01010	0010011
80004	0000000	11001	01010	000	01010	0110011
80008	0000000	00000	01010	011	01001	0000011
80012	0000000	11000	01001	001	01100	1100011
80016	0000000	00001	10110	000	10110	0010011
80020	1111111	00000	00000	000	01101	1100011

Instructions – Language of Computer
RISC-V Addressing Summary



Instructions – Language of Computer
RISC-V instruction encoding

Format	Instruction	Opcode	Funct3	Funct6/7
R-type	add	0110011	000	0000000
	sub	0110011	000	0100000
	sll	0110011	001	0000000
	xor	0110011	100	0000000
	srl	0110011	101	0000000
	sra	0110011	101	0000000
	or	0110011	110	0000000
	and	0110011	111	0000000
	lr.d	0110011	011	0001000
	sc.d	0110011	011	0001100
I-type	lb	0000011	000	n.a.
	lh	0000011	001	n.a.
	lw	0000011	010	n.a.
	lbu	0000011	100	n.a.
	lhu	0000011	101	n.a.
	addi	0010011	000	n.a.
	slli	0010011	001	000000
	xori	0010011	100	n.a.
	srli	0010011	101	000000
	srai	0010011	101	010000
	ori	0010011	110	n.a.
	andi	0010011	111	n.a.
	jalr	1100111	000	n.a.
S-type	sb	0100011	000	n.a.
	sh	0100011	001	n.a.
	sw	0100011	010	n.a.
SB-type	beq	1100111	000	n.a.
	bne	1100111	001	n.a.
	blt	1100111	100	n.a.
	bge	1100111	101	n.a.
	bltu	1100111	110	n.a.
	bgeu	1100111	111	n.a.
U-type	lui	0110111	n.a.	n.a.
UJ-type	jal	1101111	n.a.	n.a.

Instructions – Language of Computer
Decoding Machine Language

What is the assembly language statement corresponding to this machine instruction?

00578833_{hex}

The first step is converting hexadecimal to binary:

0000 0000 0101 0111 1000 1000 0011 0011

funct7	rs2	rs1	funct3	rd	opcode
0000000	00101	01111	000	10000	0110011

```
add x16, x15, x5
```

Instructions – Language of Computer

RISC-V Encoding Summary

Name (Field Size)	Field 7 bits	5 bits	5 bits	3 bits	5 bits	7 bits	Comments
R-type	funct7	rs2	rs1	funct3	rd	opcode	Arithmetic instruction format
I-type	immediate[11:0]		rs1	funct3	rd	opcode	Loads & immediate arithmetic
S-type	immed[11:5]	rs2	rs1	funct3	immed[4:0]	opcode	Stores
SB-type	immed[12,10:5]	rs2	rs1	funct3	immed[4:1,11]	opcode	Conditional branch format
UJ-type	immediate[20,10:1,11,19:12]				rd	opcode	Unconditional jump format
U-type	immediate[31:12]				rd	opcode	Upper immediate format

THANK YOU

Dr. Santhameena.S
Department of Electronics and Communication Engineering
santhameena.s@pes.edu