ENCODING OF MACHINE INSTRUCTIONS

- To be executed in a processor, an instruction must be encoded in a binary-pattern. Such encoded instructions are referred to as **Machine Instructions**.
- The instructions that use symbolic-names and acronyms are called *assembly language instructions*.
- We have seen instructions that perform operations such as add, subtract, move, shift, rotate, and branch. These instructions may use operands of different sizes, such as 32-bit and 8-bit numbers.
- Let us examine some typical cases. The instruction
- *Add R1, R2* ;Has to specify the registers R1 and R2, in addition to the OP code. If the processor has 16 registers, then four bits are needed to identify each register. Additional bits are needed to indicate that the Register addressing-mode is used for each operand.
- The instruction
- *Move 24(R0), R5* ;Requires 16 bits to denote the OP code and the two registers, and some bits to express that the source operand uses the Index addressing mode and that the index value is 24.

You

Click to add title

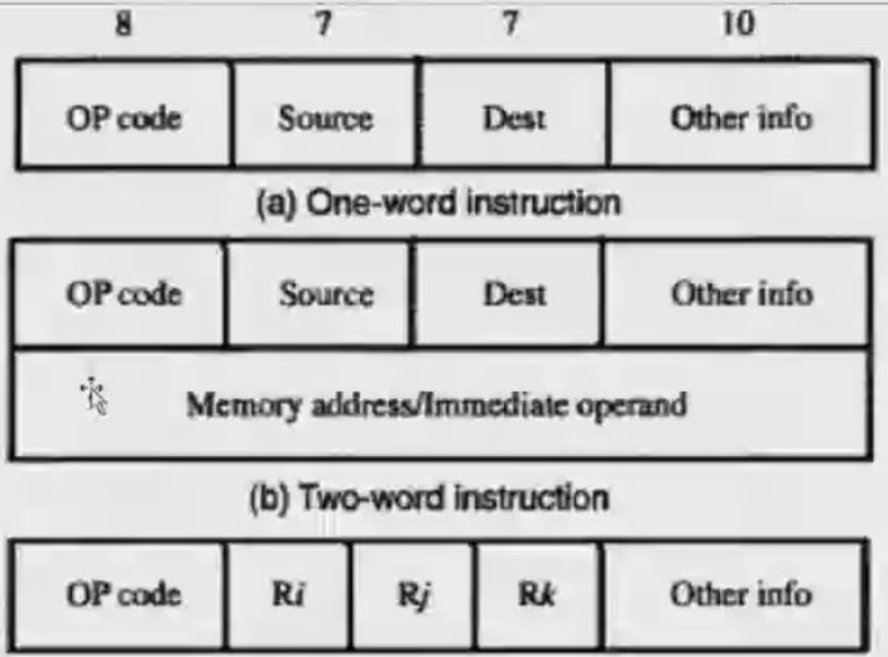


Figure 2.39 Encoding instructions into 32-bit words.

You

Click to add title

Opcode	Source	Addressing modes of source	Destination	Addressing mode of destination	Other infor
8 bits	4 bits	3 bits	4 bits	3 bits	

ADD R1,R2

You

REPRESENTATION OF NUMBERS

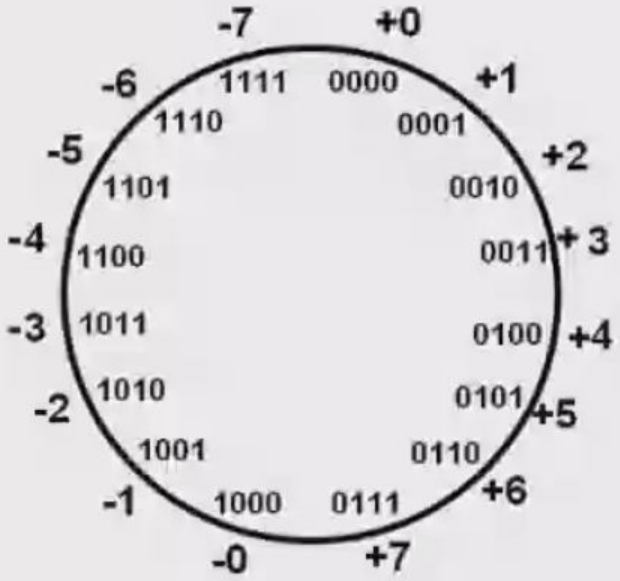
- **FIXED POINT NUMBERS**
- 31.=00110001.
- 51.=01010001.
- 0.31=.00110001
- 0.51=.01010001

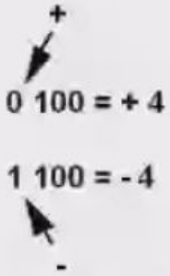
- SIGNED(NEGATIVE NUMBERS) & UNSIGNED INTEGER(+VE NUMBERS)
- SIGNED MAGNITUDE REPRESENTATION
- 1'S COMPLEMENT
- 2'S COMPLEMENT

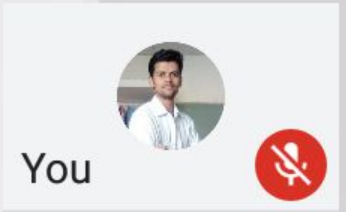
- **FLOATING POINT NUMBERS**

You

Sign and magnitude







Click to add title

- High order bit is sign: 0 = positive (or zero), 1 = negative
- Three low order bits is the magnitude:
  0 (000) thru 7 (111)
- Number range for n bits = (+/-) 2^{n-1}-1.
- Two representations for 0

B7	B6	B5	B4	B3	B2	B1	B0
SIGN			MAGNITUDE				

+6=0,000 0110
-14=1,000 1110
+24=0,001 1000
-64=1,100 0000
+VE
MIN num is 0,000 0000
Max num is 0,111 1111=+127
Max –ve number is 1,111 1111=-127

You

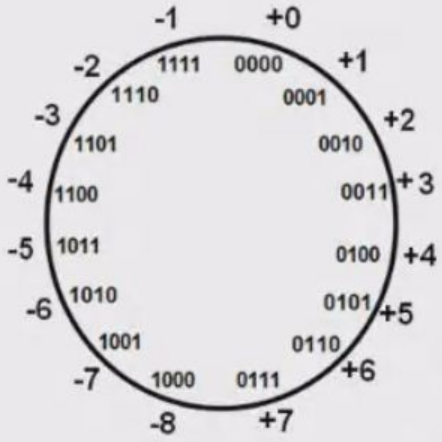
Click to add title

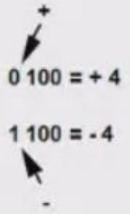
- 2 representation of 0
- +0=0,000 0000
- -0=1,000 0000

You

Two's complement

like 1's comp except shifted one position clockwise





Notation	Min	Max
Unsigned:	0	255
One's Comp:	-127	+127
Two's Comp:	-128	+127

You

Click to add title

- +6→0,00110
- -9→1,10110
- 1,11100→1's comp→0,00011

You

Click to add title

(a)	0 0 1 0	(+2)	(b)	0 1 0 0	(+4)
	+ 0 0 1 1	(+3)		+ 1 0 1 0	(−6)
	0 1 0 1	(+5)		1 1 1 0	(−2)
(c)	1 0 1 1	(−5)	(d)	0 1 1 1	(+7)
	+ 1 1 1 0	(−2)		+ 1 1 0 1	(−3)
	1 0 0 1	(−7)		0 1 0 0	(+4)
(e)	1 1 0 1	(−3)		1 1 0 1	
	− 1 0 0 1	(−7)	⇒	+ 0 1 1 1	
				0 1 0 0	(+4)
(f)	0 0 1 0	(+2)		0 0 1 0	
	− 0 1 0 0	(+4)	⇒	+ 1 1 0 0	
				1 1 1 0	(−2)
(g)	0 1 1 0	(+6)		0 1 1 0	
	− 0 0 1 1	(+3)	⇒	+ 1 1 0 1	
				0 0 1 1	(+3)
(h)	1 0 0 1	(−7)		1 0 0 1	
	− 1 0 1 1	(−5)	⇒	+ 0 1 0 1	
				1 1 1 0	(−2)
(i)	1 0 0 1	(−7)		1 0 0 1	
	− 0 0 0 1	(+1)	⇒	+ 1 1 1 1	
				1 0 0 0	(−8)
(j)	0 0 1 0	(+2)		0 0 1 0	
	− 1 1 0 1	(−3)	⇒	+ 0 0 1 1	
				0 1 0 1	(+5)

Figure 1.6 2's-complement Add and Subtract operations.

You

OVERFLOW IN INTEGER ARITHMETIC

- When result of an arithmetic operation is outside the representable-range, an **arithmetic overflow**
- is said to occur.
- For example: If we add two numbers +7 and +4, then the output sum S is 1011(ß0111+0100), which is the code for -5, an incorrect result.
- An overflow occurs in following 2 cases
- Overflow can occur only when adding two numbers that have the same sign.
- The carry-out signal from the sign-bit position is not a sufficient indicator of overflow when adding signed numbers.