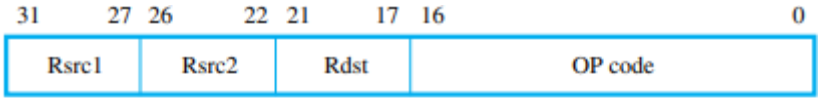
Encoding of Machine Instructions:

we have introduced a variety of useful instructions and addressing modes. We have used a generic form of assembly language to emphasize basic concepts without relying on processor-specific acronyms or mnemonics. Assembly- language instructions symbolically express the actions that must be performed by the processor circuitry. To be executed in a processor, assembly-language instructions must be converted by the assembler program, in to machine instructions that are encoded in a compact binary pattern.

Let us now examine how machine instructionsmay be formed.The Add instruction

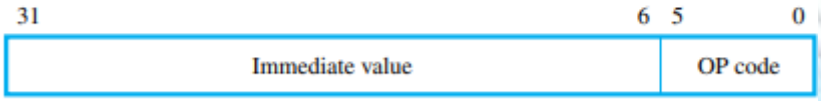
Add Rdst, Rsrc1, Rsrc2

The above instruction representative of a class of three-operand instructions that use operands in processor registers. Registers Rdst, Rsrc1, and Rsrc2 hold the destination and two source operands. If a processor has 32 registers, then itis necessary to use five bits to specify each of the three registers in such instructions. If each instruction is implemented in a 32-bit word, the remaining 17 bits can be used to specify the OP code that indicates the operation to be performed



(a) Register-operand format

(b) Immediate-operand format



(c) Call format

Possible instruction formats.

Now consider instructions in which one operand is given using the Immediateaddressing mode, such as AddRdst, Rsrc, #Value

Of the 32 bits available, ten bits are needed to specify the two registers. The remaining 22 bits must give the OP code and the value of the immediate operand. The most useful sizes of immediate operands are 32, 16, and 8 bits. Since 32 bits are not available, a good choice is to allocate 16 bits for the immediate operand. This leaves six bits for specifying the OP code.

This format can also be used for Load and Store instructions, where the Index addressing mode uses the 16-bit field to specify the offset that is added to the contents of the index register.

The format in Figure b can also be used to encode the Branch instructions. The Branch-greater-thaninstruction at memory addresses 128.

BGT R2, R0, LOOP

if the contents of register R0 are zero. The registers R2 and R0 can be specified inthe two register fields in Figure b. The six-bit OP code has to identify the BGT operation. The 16-bit immediate field can be used to provide the information needed to determine the branch target address, which is the location of theinstruction with the label LOOP. The target address generally comprises 32 bits. Since there is no space for 32 bits, the BGT instruction makes use of the immediate field to give an offset from the location of this instruction in the

program to the required branch target. At the time the BGT instruction is beingexecuted, the program counter, PC, has been incremented to point to the next instruction, which is the Store instruction at address

132. Therefore, the branch offset is 132 − 112 = 20. Since the processor computes the target address by adding the current contents of the PC and the branch offset, the required offset in this example is negative, namely −20. Finally, we should consider the Call instruction, which is used to call a subroutine. It only needs to specify the OP code and an immediate value that is used to determine the address of the first instruction inthe subroutine. If six bits are used for the OP code, then the remaining 26 bits can be used to denote the immediate value. This gives the format shown in c.