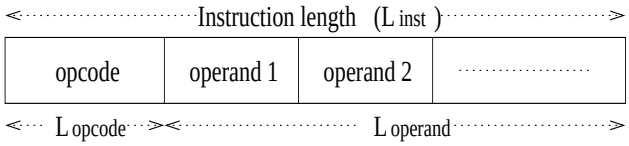
Lecture 7-8: February 26, 2021
Computer Architecture and Organization-I
Biplab K Sikdar

0.4 Opcode Optimization



Width of opcode field depends on the number of instructions in the instruction set.
Opcode can be of fixed length -that is, fixed number of bits for opcode field.
In fixed format, for n-bit opcode, there can be 2^n instructions in instruction set.
However, reducing n cannot always be a realistic solution.
The alternative: choice of variable length opcode.

Opcode extension was introduced in m/c of old days. Example: DEC PDP-8.
Instruction length is 12-bit (Figure 10).

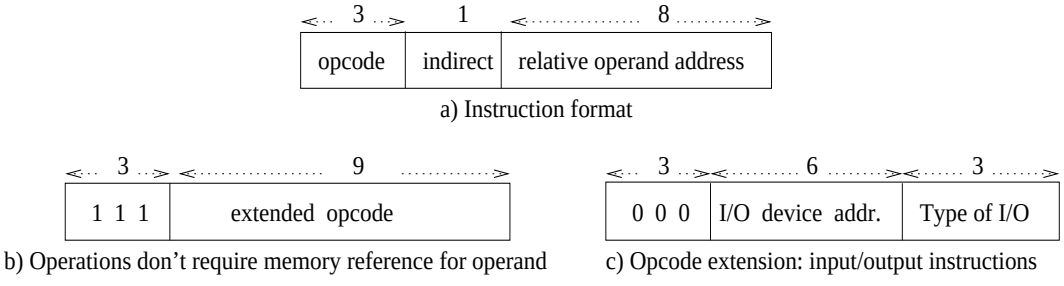
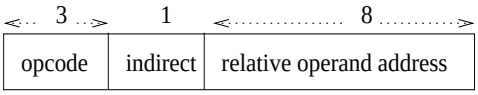
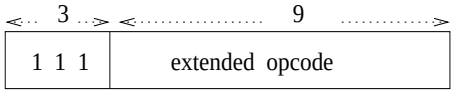


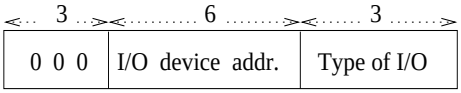
Figure 10: Instruction format of PDP-8



a) Instruction format



b) Operations don't require memory reference for operand



c) Opcode extension: input/output instructions

In normal representation, 3-bit is specified for opcode, 1-bit is for direct or indirect addressing and 8-bit is for relative operand (memory) address.

Out of eight unique patterns in 3-bit opcode (Figure 10(a)), six patterns (001 to 110) are for opcode, each refers to a single-address instruction.

If the first 3-bit of the instruction is 111, then next 9-bit of the instruction specifies extended opcode (Figure 10(b)).

An instruction without any explicit operand can be encoded with this 9-bit.

Example : CLEAR (clear AC), Skip, Right shift/Left shift, STOP etc.

If the first 3-bit is set to 000, then the next 6 bits specify I/O device address and least significant 3 bits define type of I/O operation (Figure 10(c)).

Data dependent opcode Here tag bits are added with memory words.

In CISC m/c, there are separate integer arithmetic unit and floating point unit.

Such m/c includes multiple instruction for an arithmetic operation. Example - ADD can be defined as ADDinteger, ADDreal and ADDfloating. That is, three ADD instructions are to be included in the instruction set.

In data dependent opcode scheme, these three ADD is replaced by a single addition instruction. Tag of operands specify the type of operands. Example:

$$\text{ADD } r_1, r_2$$

If r_1 and r_2 are integer, the ADD operation is sent to integer arithmetic unit.

In RISC, use of tag is necessary as RISC considers a very few instructions.

Frequency dependent opcode Some instructions are used frequently in the programs. Example: MOVE, shift (RSHIFT/LSHIFT) etc.

If frequently used instructions are represented in fewer number words, the size of a program can be reduced.

In frequency dependent opcode, the most frequently used instructions are encoded with lesser number of bits. Example: in 8085 microprocessor, MOV R1, R2 is an one byte (one word) instruction. The MOV has 2-bit opcode. The rest 6 bits are to denote two register operands R1 (3-bit), and R2 (3-bit).

MOV	R1,	R2
xx	xxx	xxx

However, opcode of LDA/STA is of 8 bits. These two are 3-byte instructions.

To define frequency dependent opcode, a solution can be Hoffman encoding.

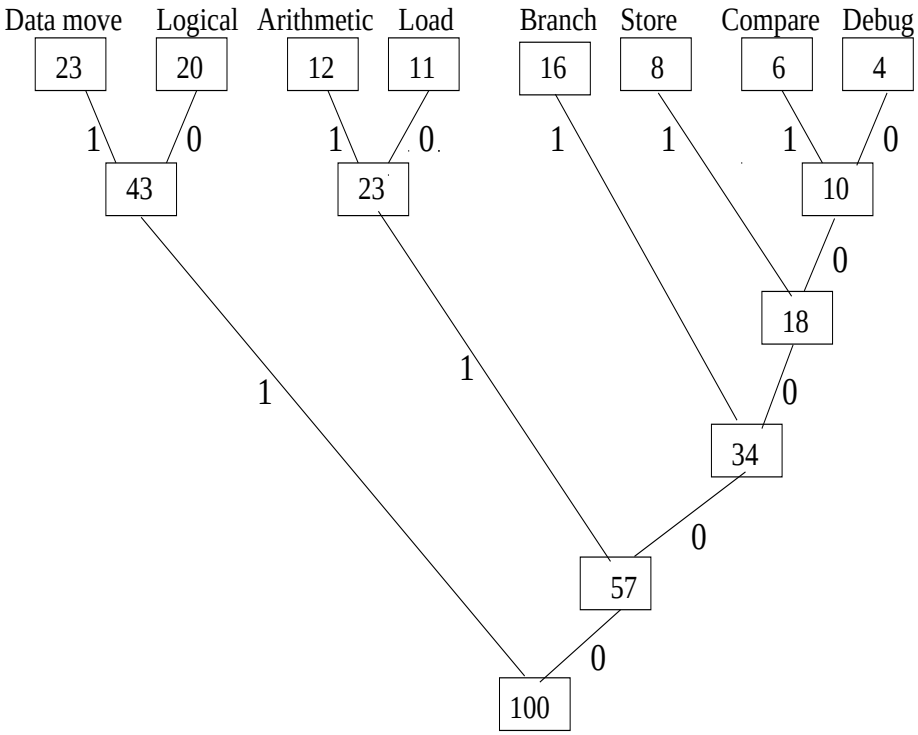


Figure 11: Frequency dependent opcode optimization

Figure 11 describes Hoffman encoding for a m/c with eight type of instrctions.

Now, to encode instructions, let assume left child is 1 and right is 0.

A trace from root to leaf defines opcode for the instruction defined by the leaf.

Decoding of variable length opcodes is difficult than that of fixed length opcode.

Number of bits of opcode to be transferred to IR from DR depends on opcode type.

RISC and superscalar computers implement fixed-length instruction format.

0.7 Instruction Implementation

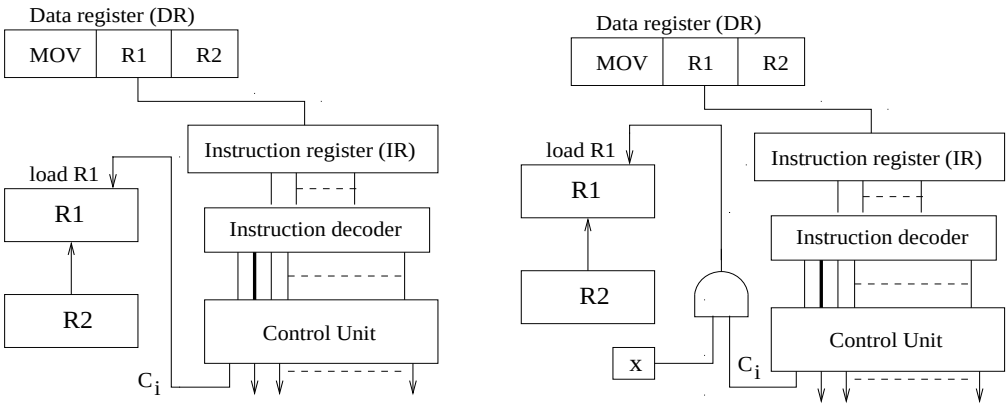
Execution of an (macro) instruction involves execution of a sequence of micro-operations.

0.7.1 Data movement

Here the sources and destinations can be the register or memory.

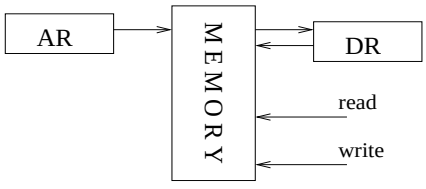
Implementation of register to register data transfer instruction is shown in Figure 12(a).

MOV R1, R2 (R1 ← R2)



(i) MOV R1, R2 (ii) If x then MOV R1, R2

a) Register to register data transfer



read : DR ← M(AR), memory to register transfer
write : M(AR) ← DR, register to memory transfer

b) Data transfer between register and memory

Figure 12: Data transfer instruction implementation

However, the conditional data movement

$$\text{if } x \text{ MOV R1, R2} \qquad (\text{R1} \leftarrow \text{R2})$$

is realized (as shown in Figure 12(a)(ii)).

The execution of data movement instruction like

MOV A1, A2

where A1/A2 are memory addresses, implies completion of micro-operations

AR ← A2	this is a register to register data transfer
DR ← M(AR)	memory to register transfer
AR ← A1	register to register transfer
M(AR) ← DR	register to memory data transfer

LOAD: memory to register data transfer.

STORE: executes register to memory transfer.

Data movement instruction execution requires bus transfers (Figure 13).

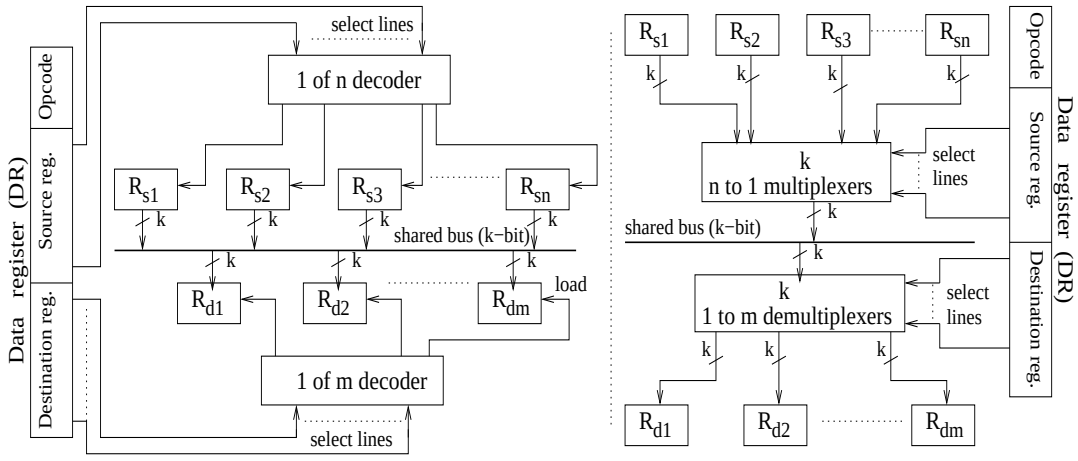


Figure 13: Data transfer from many sources to many destinations

Data movement from R_s to R_d: implies data should be transferred from R_s to bus and then from bus to R_d.

Figure 13 shows data transfer from one of many sources to one of many destinations.

Figure 14: implementation while sources and destinations are same set of registers.

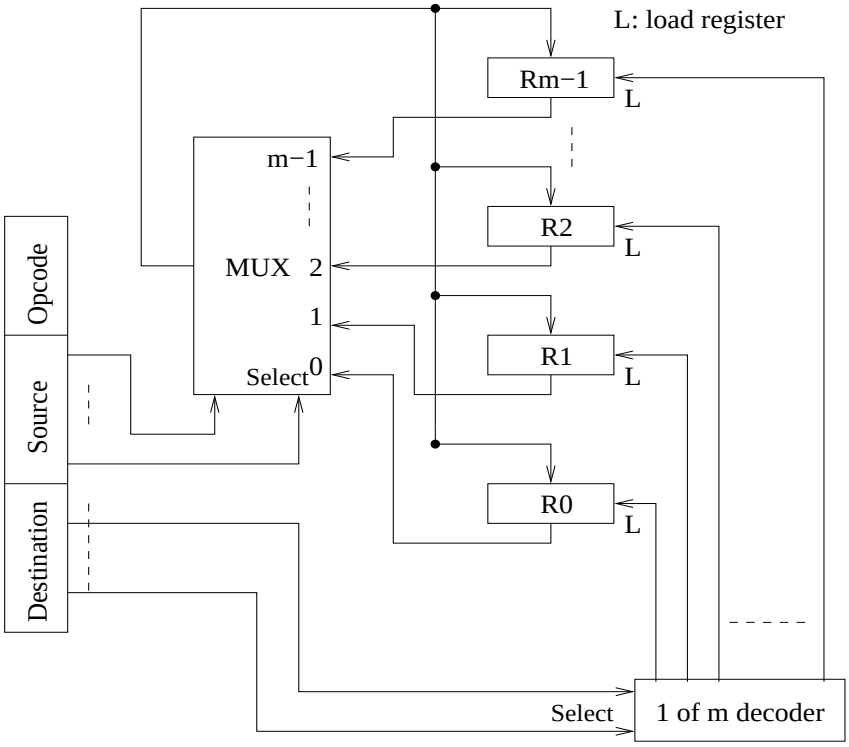


Figure 14: Data transfer among same set of registers

0.7.2 Branch control

The conditional branch, unconditional branch, skip, jump to subroutine etc.

Unconditional branch can be implemented by execution of register (DR) to register (PC) transfer micro-operation (Figure 15) -

$$PC \leftarrow DR \text{ (operand)}.$$

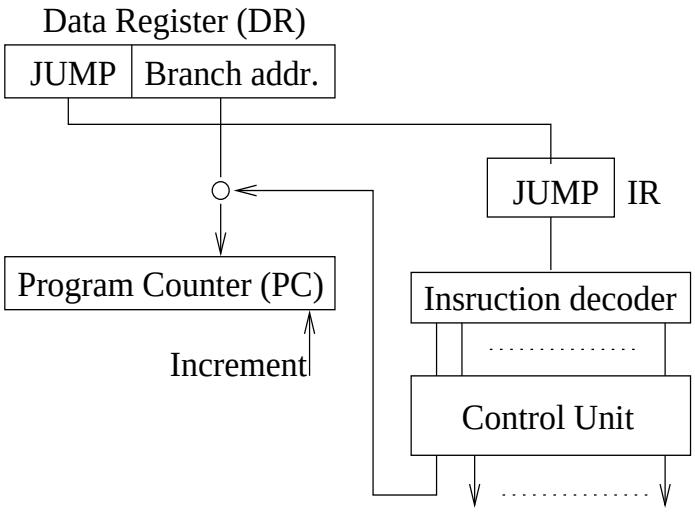


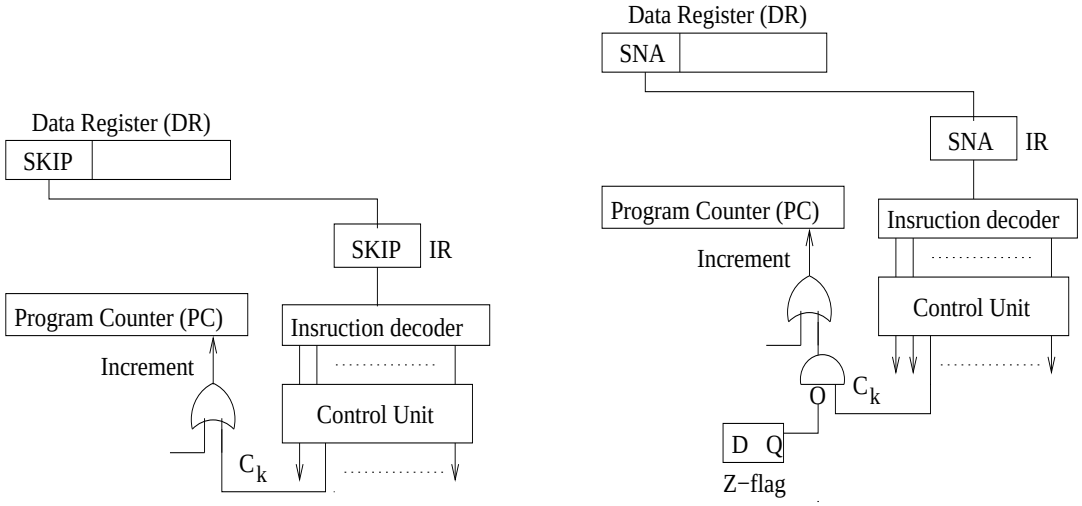
Figure 15: Branch control

Skip implies one instruction is to be skipped.

	⋮
L:	SKIP
L+1:	ADD X, Y, Z
L+2:	I_i
	⋮

ADD instruction will be skipped and next executable instruction will be I_i.

Implementation of unconditional/conditional SKIP is shown in Figure 16.



a) Unconditional skip instruction implementation

b) Conditional skip instruction implementation

Figure 16: Skip instruction implementation

Conditional branch The conditions (jump on zero/non-zero, jump on positive/negative, jump on parity, jump on carry, jump on even/odd, jump on overflow etc) can be tested from flags (Figure 17).

If condition is satified, then register to register to data transfer

$$PC \leftarrow DR \text{ (operand)}.$$

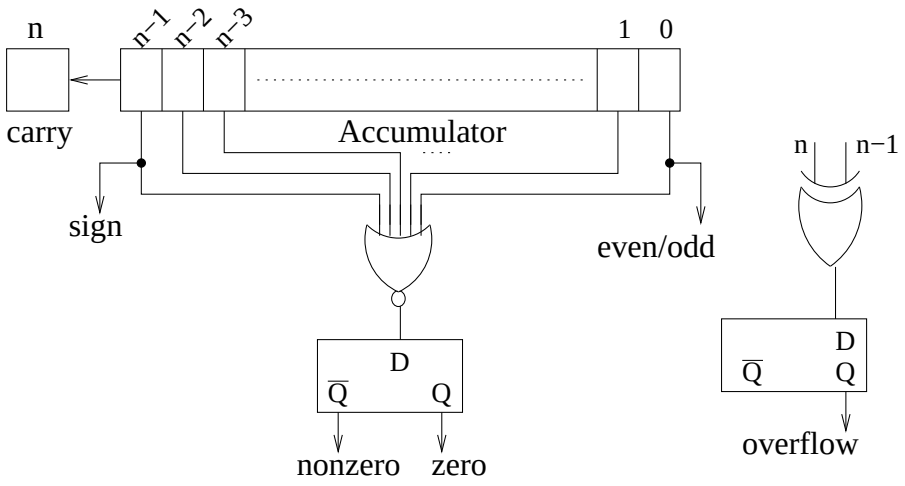


Figure 17: Setting condition flags

Conditional Skip: one instruction is to be skipped if condition is satiesfied.

Implementation of SNA (skip on non-zero AC) is shown in Figure 16(b).