InstructionSetArchitecture

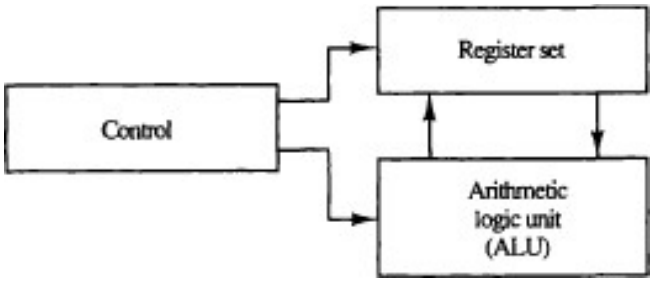
Central processing unit:

INTRODUCTION

The part of the computer that performs the bulk of data-processing operations is called thecentralprocessingunitandisreferredtoastheCPU.TheCPUismadeupofthreemajorparts,as shown in Fig. The register set stores intermediate data used during the execution of the instructions. The arithmetic logic unit (ALU) performs the required micro operations forexecutingtheinstructions.Thecontrolunitsupervisesthetransferofinformationamongtheregister sandinstructsthe ALU as to which operation to perform.

The CPU performs a variety of functions dictated by the type of instructions that are incorporated in the computer. Computer architecture is sometimes defined as the computer structure and behavior as seen by the programmer that uses machine language instructions. This includes the instruction formats, addressing modes, the instruction set, and the general organization of the CPU registers.

Components of CPU



General register organization

A bus organization for seven CPU registers is shown in Fig. The output of each register is connected to two multiplexers (MUX) to form the two buses A and B . The selection lines in each multiplexer select one register or the input data for the particular bus.

The A and B buses form the inputs to a common arithmetic logic unit (ALU). The operations elected in the ALU determines the arithmetic or logic micro operation that is to be performed. The result of the micro operation is available for output data and also goes into the inputs of all the registers. The register that receives the information from the output bus is selected by a decoder. The decoder activates one of the register load inputs, thus providing a transfer path between the data in the output bus and the inputs of the selected destination register.

The control unit that operates the CPU bus system directs the information flow through theregistersandALUbyselectingthevariouscomponentsinthesystem.Forexample,to perform

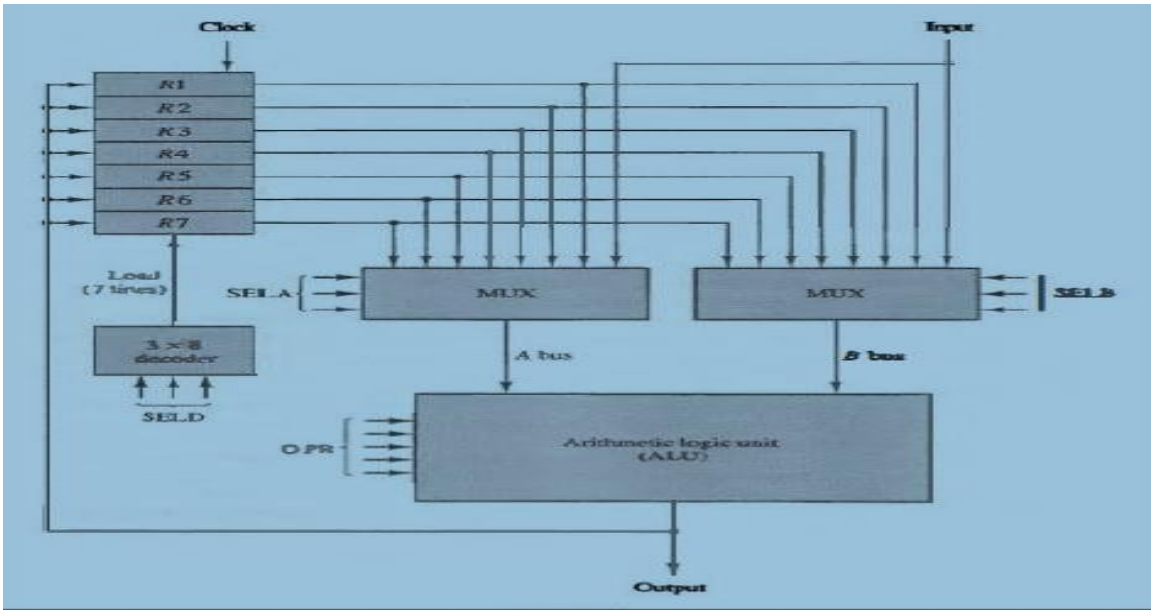
the operation R 1 <--R2 + R3 the control must provide binary selection variables to the following selector inputs:

1. MUX A selector (SELA): to place the content of R2 into bus A .
2. MUX B selector (SELB): to place the content o f R 3 into bus B .
3 . ALU operation selector (OPR): to provide the arithmetic addition A + B .
4. Decoder destination selector (SELD): to transfer the content of the output bus into R1

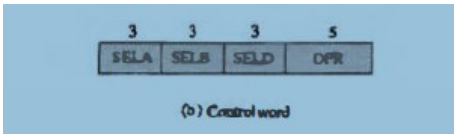
.

The four control selection variables are generated in the control unit and must be available at the beginning of a clock cycle. The data from the two source registers propagate through the gates in the multiplexers and the ALU, to the output bus, and into the inputs of the destination register, all during the clock cycle interval. Then, when the next clock transition occurs, the binary information from the output bus is transferred into R 1.

To achieve a fast response time, the ALU is constructed with high speed circuits.



Control Word



There are 14 binary selection inputs in the unit, and their combined value specifies a control word .Encoding of registers election fields

Binary Code	SELA	SELB	SELD
000	Input	Input	None
001	R1	R1	R1
010	R2	R2	R2
011	R3	R3	R3
100	R4	R4	R4
101	R5	R5	R5
110	R6	R6	R6
111	R7	R7	R7

ALU

The ALU provides arithmetic and logic operations. In addition, the CPU must provide shiftoperations.TheshiftermaybeplacedintheinputoftheALUtoprovideapreshiftcapability,or at the output of the ALU to provide post shifting capability. In some cases, the shift operations are included with the ALU.

OPR Select	Operation	Symbol
00000	Transfer A	TSFA
00001	Increment A	INCA
00010	Add A + B	ADD
00101	Subtract A − B	SUB
00110	Decrement A	DECA
01000	AND A and B	AND
01010	OR A and B	OR
01100	XOR A and B	XOR
01110	Complement A	COMA
10000	Shift right A	SHRA
11000	Shift left A	SHLA

Examples of Micro operations R

1 <- R 2 - R3

specifies R2 for the A input of the ALU, R3 for the B input of the ALU, R1 for the destination register, and an ALU operation to subtract A -B.

Field:	SELA	SELB	SELD	OPR
Symbol:	R2	R3	R1	SUB
Control word:	010	011	001	00101

	Symbolic Designation				
Microoperation	SELA	SELB	SELD	OPR	Control Word
$R1 \leftarrow R2 - R3$	R2	R3	R1	SUB	010 011 001 00101
$R4 \leftarrow R4 \vee R5$	R4	R5	R4	OR	100 101 100 01010
$R6 \leftarrow R6 + 1$	R6	—	R6	INCA	110 000 110 00001
$R7 \leftarrow R1$	R1	—	R7	TSFA	001 000 111 00000
Output $\leftarrow R2$	R2	—	None	TSFA	010 000 000 00000
Output $\leftarrow$ Input	Input	—	None	TSFA	000 000 000 00000
$R4 \leftarrow$ shl $R4$	R4	—	R4	SHLA	100 000 100 11000
$R5 \leftarrow 0$	R5	R5	R5	XOR	101 101 101 01100

InstructionFormats

The most common fields found in instruction formats are :

1. An operation code field that specifies the operation to be performed.
2. An address field that designates a memory address or a process or register.
3. A mode field that specifies the way the operand or the effective address is determined.

Most computers fall in to one of three types of CPU organizations:

1. Single accumulator organization.
2. General register organization.
3. Stack organization

To illustrate the influence of the number of addresses on computer programs, we will evaluate the arithmetic statement

X=(A+B)•(C+D)

Using zero, one, two, or three address instructions.

Three-Address Instructions

Computerswiththree-addressinstructionformatscanuseeachaddressfieldtospecifyeitheraprocessorregisteroramemoryoperand.TheprograminassemblylanguagethatevaluatesX=(A+B)•(C+D)isshownbelow,togetherwithcommentsthatexplaintheregistertransferoperationofeachinstruction.

```
ADD   R1, A, B     R1←M[A] + M[B]
ADD   R2, C, D     R2←M[C] + M[D]
MUL   X, R1, R2    M[X]←R1*R2
```

It is assumed that the computer has two processor registers, R 1 and R2. The symbol M [A]denotes the operand at memory address symbolized by A . The advantage o f the three-address format i s that i t results in short programs when evaluating arithmetic expressions. The disadvantage is that the binary-coded instructions require too many bits to specify three addresses.

An example of a commercial computer that uses three-address instructions is the Cyber 170.The instruction formats in the Cyber computer are restricted to either three register address fields or two register address fields and one memory address field.

Two-Address Instructions

Two-address instructions are the most common in commercial computers. Here again each address field can specify either a processor register or a memory word. The program to evaluate $X = (A + B) * (C + D)$ is as follows:

```
MOV    R1, A     R1 ← M[A]
ADD    R1, B     R1 ← R1 + M[B]
MOV    R2, C     R2 ← M[C]
ADD    R2, D     R2 ← R2 + M[D]
MUL    R1,R2     R1 ← R1 * R2
MOV    X, R1     M[X] ← R1
```

The MOV instruction moves or transfers the operands to and from memory and processor registers. The first symbol listed in an instruction is assumed to be both a source and the destination where the result of the operation is transferred.

One-Address Instructions

One-address instructions use an implied accumulator (*AC*) register for all data manipulation. For multiplication and division there is a need for a second register. However, here we will neglect the second register and assume that the *AC* contains the result of all operations. The program to evaluate $X = (A + B) * (C + D)$ is

```
LOAD     A     AC ← M[A]
ADD      B     AC ← AC + M[B]
STORE    T     M[T] ← AC
LOAD     C     AC ← M[C]
ADD      D     AC ← AC + M[D]
MUL      T     AC ← AC * M[T]
STORE    X     M[X] ← AC
```

All operations are done between the *AC* register and a memory operand. *T* is the address of a temporary memory location required for storing the intermediate result.

Zero-Address Instructions

A stack-organized computer does not use an address field for the instructions ADD and MUL. The PUSH and POP instructions, however, need an address field to specify the operand that communicates with the stack. The following program shows how $X = (A + B) * (C + D)$ will be written for a stack-organized computer. (*TOS* stands for top of stack.)

```
PUSH    A    TOS ← A
PUSH    B    TOS ← B
ADD          TOS ← (A + B)
PUSH    C    TOS ← C
PUSH    D    TOS ← D
ADD          TOS ← (C + D)
MUL          TOS ← (C + D) * (A + B)
POP     X    M[X] ← TOS
```

To evaluate arithmetic expressions in a stack computer, it is necessary to convert the expression into reverse Polish notation. The name "zero-address" is given to this type of computer because of the absence of an address field in the computational instructions.