www.teachics.org

Basic Computer Organization and Design

Computer Organization & Architecture - MODULE 3
PART 1

Instruction Codes

Instruction Codes

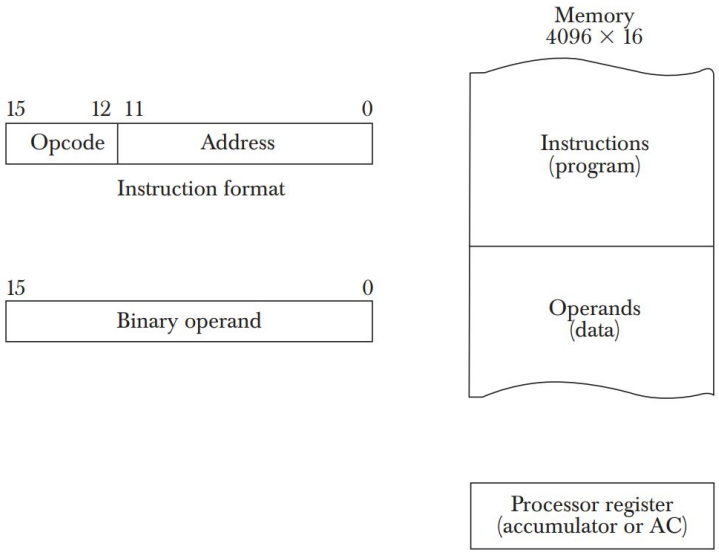
- The organization of the computer is defined by its internal registers, the timing and control structure, and the set of instructions that it uses.
- A **computer instruction** is a binary code that specifies a sequence of microoperations for the computer.
- An **instruction code** is a group of bits that instruct the computer to perform a specific operation.
- Instruction code is usually divided into two parts.
    - **Operation part** - Group of bits that define such operations as add, subtract, multiply, shift, and complement.
    - **Address part** - Contains registers or memory words where the address of operand is found or the result is to be stored.
- Each computer has its own instruction code format.

Operation Code

- The **operation code(op-code)** of an instruction is a group of bits that define such operations as add, subtract, multiply, shift, and complement.
- The number of bits required for the operation code of an instruction depends on the total number of operations available in the computer.(n bits for 2^n operations)
- An operation code is sometimes called a **macro-operation** because it specifies a set of micro-operations.

Stored Program Organization

- Simplest way to organize computer is to have one processor register(Accumulator AC) and an instruction code format with two parts.
  - **First**-Operation to be performed
  - **Second** – Address
- The memory address tells the control where to find an operand in memory.
- This operand is read from memory and used as the data to be operated on together with the data stored in the processor register.



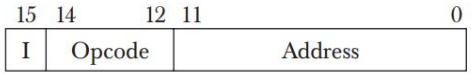
- Instructions are stored in one section of the memory and data in another.
- For a memory unit with 4096 words we need 12 bits to specify an address since 2^{12}=4096.
- 4 bits are available for opcode to specify one out of 16 possible operations.

Steps

- The control reads a 16-bit instruction from the program portion of memory.
- It uses the 12-bit address part of the instruction to read a 16-bit operand from the data portion of memory.
- It then executes the operation specified by the operation code.
- The operation is performed with the memory operand and the content of AC.

Direct & Indirect Addressing Modes

- Following **Addressing Modes** are used for address portion of the instruction code.
  - **Immediate**- The address part specifies an operand.
    Eg: ADD 5
  - **Direct**- The address part specifies the address of an operand.
  - **Indirect**- The address part specifies a pointer(another address) where the address of the operand can be found.
- One bit of the instruction code(**I**) can be used to distinguish between a direct and an indirect address.

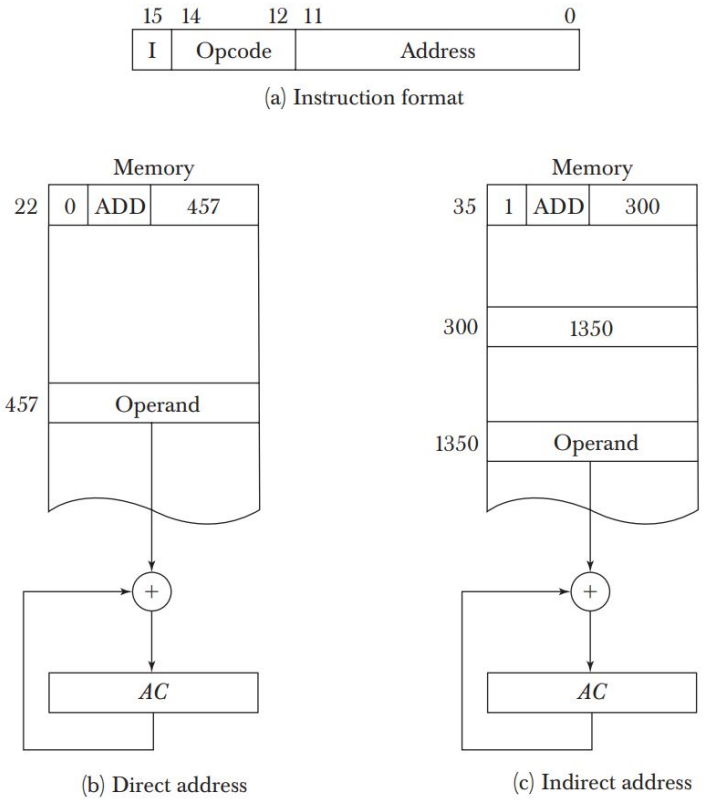


(a) Instruction format

Effective Address

It is the address of the operand in a computation-type instruction or the target address in a branch-type instruction.

The effective address in the instruction of Fig.(b) is **457** and in the instruction of Fig(c) is **1350**.



Computer Registers

Computer Registers

Need of Registers?

- Computer instructions are normally stored in consecutive memory locations and are executed sequentially one at a time.
- The control reads an instruction from a specific address in memory and executes it. It then continues by reading the next instruction in sequence and executes it, and so on.
- This type of instruction sequencing needs a **counter** to calculate the address of the next instruction after execution of the current instruction is completed.
- It is also necessary to provide a register in the control unit for storing the **instruction code** after it is read from memory.
- The computer needs **processor registers** for manipulating data and a register for holding a memory **address**.

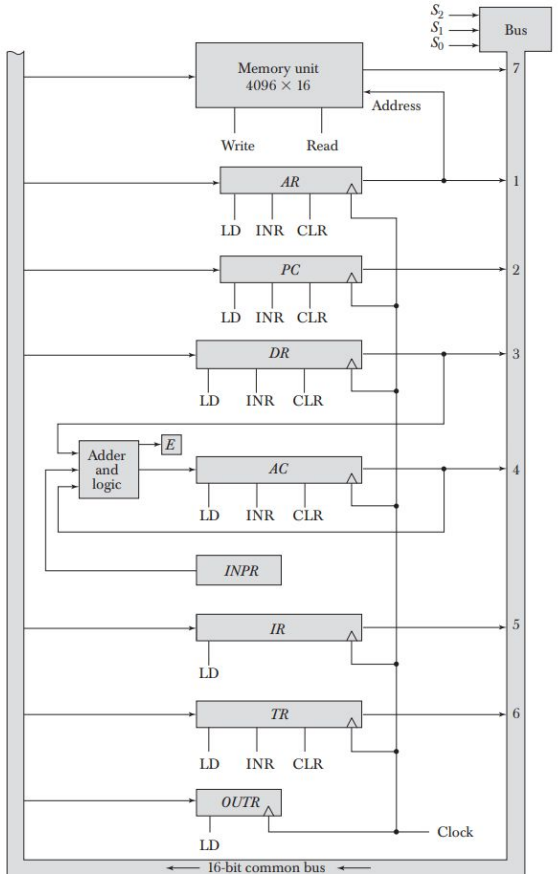
List of basic Registers

Code	Bits	Name	Purpose
DR	16	Data Register	Holds memory operand
AR	12	Address register	Holds address for memory
AC	16	Accumulator	Processor register
IR	16	Instruction register	Holds instruction code
PC	12	Program counter	Holds address of instruction
TR	16	Temporary register	Holds temporary data
INPR	8	Input register	Holds input character
OUTR	8	Output register	Holds output character

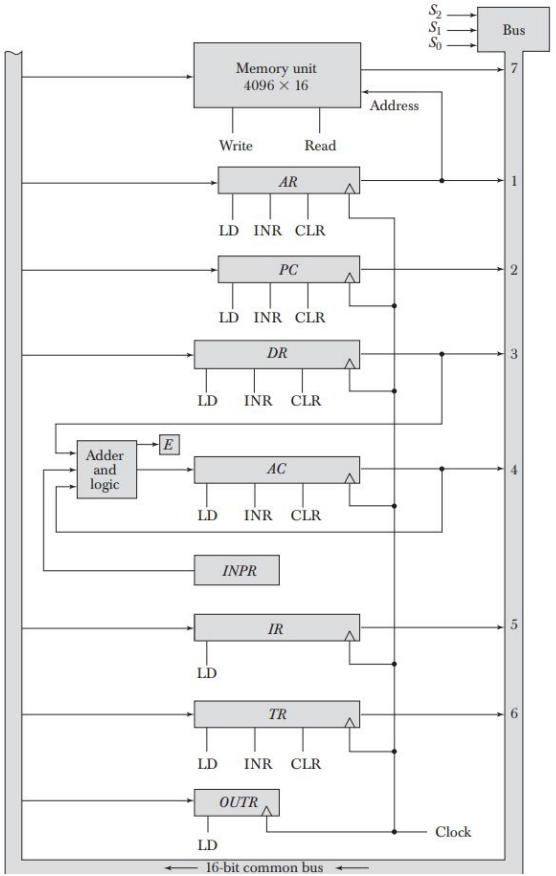
Common Bus System

Need of Common Bus System ?

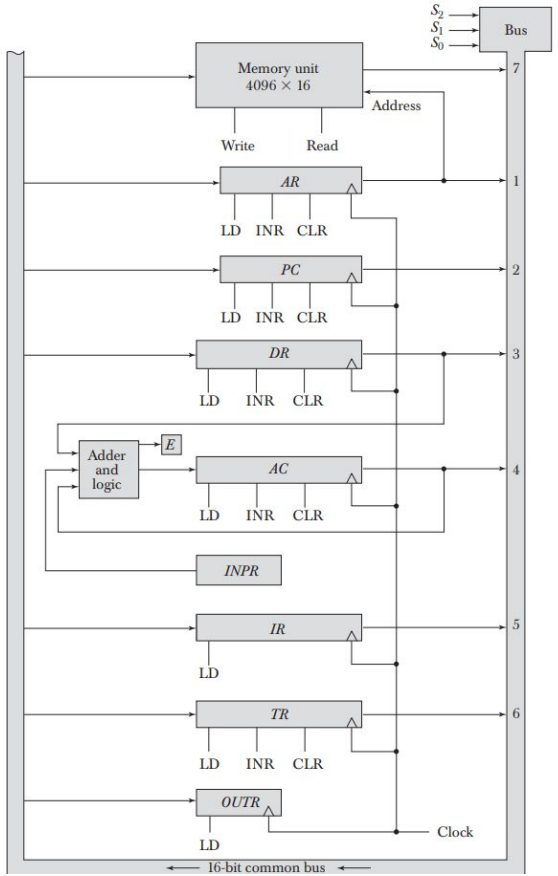
- The basic computer has eight registers, a memory unit, and a control unit.
- Paths must be provided to transfer information from one register to another and between memory and registers.
- The number of wires will be excessive if connections are made between the outputs of each register and the inputs of the other registers.
- Hence more efficient scheme with a common bus is used.



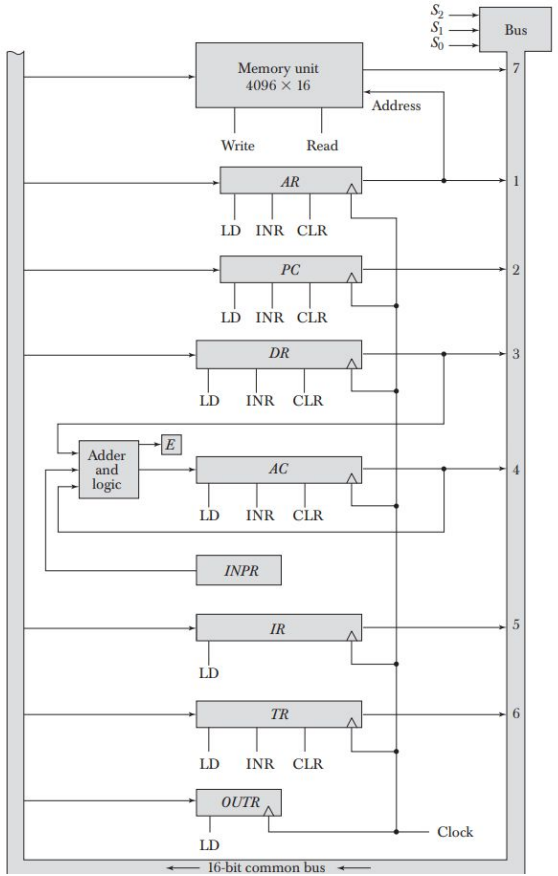
- The outputs of seven registers and memory are connected to the common bus.
- The specific output that is selected for the bus lines at any given time is determined from the binary value of the selection variables S_2, S_1, and S_0.
- The number along each output shows the decimal equivalent of the required binary selection.
- For example, the number along the output of DR is 3. The 16-bit outputs of DR are placed on the bus lines when $S_2S_1S_0$ =011 since this is the binary value of decimal 3.



- The lines from the common bus are connected to the inputs of each register and the data inputs of the memory.
- The particular register whose LD (load) input is enabled receives the data from the bus during the next clock pulse transition.
- The memory receives the contents of the bus when its write input is activated.
- The memory places its 16-bit output onto the bus when the read input is activated and $S_2S_1S_0$ =111.



- INPR is connected to provide information to the bus but OUTR can only receive information from the bus.
- This is because INPR receives a character from an input device which is then transferred to AC.
- OUTR receives a character from AC and delivers it to an output device.
- There is no transfer from OUTR to any of the other registers.



- The inputs of AC come from an adder and logic circuit.
- This circuit has three sets of inputs.
- One set of 16-bit inputs come from the outputs of AC. They are used to implement register micro-operations such as complement AC and shift AC.
- Another set of 16-bit inputs come from the data regisler DR. The inputs from DR and AC are used for arithmetic and logic microoperations.
- A third set of 8-bit inputs come from the input register INPR.

Computer Instructions

Instruction Format

- The basic computer has three instruction code formats each having 16 bits
  - Memory reference instructions
  - Register reference instructions
  - I/O instructions
- The opcode part of the instruction contains three bits and the meaning of the remaining 13 bits depends on the operation code encountered.

Memory reference instructions

- Bits 0-11 for specifying address.
- Bits 12-14 for specifying address.
- 15th bit specifies addressing modes. (0 for direct and 1 for indirect)

- Opcode=000 through 110
- I=0 or 1
- Eg:
  - AND - 0xxx(direct) or 8xxx(indirect)
  - ADD - 1xxx or 9xxx

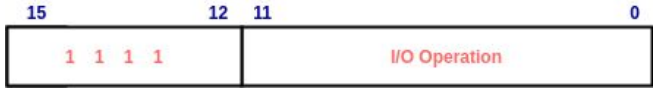
Register reference instructions

- Recognized by the operation code 111 with a 0 in the 15th bit of the instruction.
- Specifies an operation on or a test of the AC register.
- An operand from memory is not needed.
- Therefore the 12 bits are used to specify the operation to be executed.

- Eg:
  - CLA - 7800 : Clear AC
  - CLE - 7400 : Clear E

I/O instructions

- These instructions are needed for transfering informations to and from AC register.
- Recognized by the opcode 111 and a 1 in the 15th bit.
- Bits 0-11 specify the type of I/O Operation performed.



- Eg:
  - INP - F800 : Input characters to AC
  - OUT - F400 : Output characters from AC

Symbol	Hexadecimal code		Description
	$I = 0$	$I = 1$	
AND	0xxx	8xxx	AND memory word to *AC*
ADD	lxxx	9xxx	Add memory word to *AC*
LDA	2xxx	Axxx	Load memory word to *AC*
STA	3xxx	Bxxx	Store content of *AC* in memory
BUN	4xxx	Cxxx	Branch unconditionally
BSA	5xxx	Dxxx	Branch and save return address
ISZ	6xxx	Exxx	Increment and skip if zero
CLA	7800		Clear *AC*
CLE	7400		Clear *E*
CMA	7200		Complement *AC*
CME	7100		Complement *E*
CIR	7080		Circulate right *AC* and *E*
CIL	7040		Circulate left *AC* and *E*
INC	7020		Increment *AC*
SPA	7010		Skip next instruction if *AC* positive
SNA	7008		Skip next instruction if *AC* negative
SZA	7004		Skip next instruction if *AC* zero
SZE	7002		Skip next instruction if *E* is 0
HLT	7001		Halt computer
INP	F800		Input character to *AC*
OUT	F400		Output character from *AC*
SKI	F200		Skip on input flag
SKO	F100		Skip on output flag
ION	F080		Interrupt on
IOF	F040		Interrupt off

Instruction Set Completeness

- The set of instructions are said to be complete if the computer includes a sufficient number of instructions in each of the following categories
  a. Arithmetic, logical, and shift instructions
  b. Instructions for moving information to and from memory and processor registers
  c. Program control instructions
  d. Input and output instructions

Timing and Control

Timing and Control

- The timing for all registers in the basic computer is controlled by a master clock generator.
- The clock pulses are applied to all flip-flops and registers.
- The clock pulses do not change the state of a register unless the register is enabled by a control signal.
- Two major types of control organization:
  - hardwired control
  - microprogrammed control.

Hardwired Control

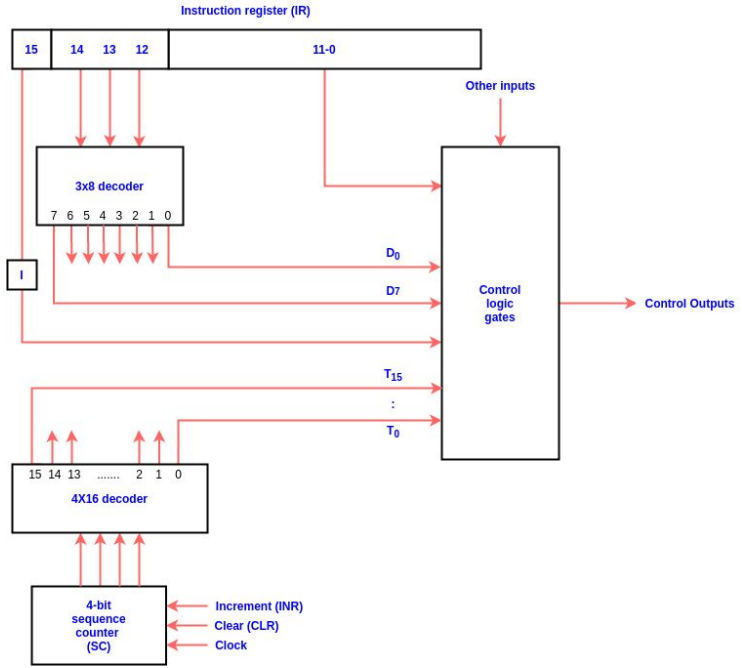
- The control logic is implemented with gates, flip-flops, decoders, and other digital circuits.
- It can be optimized to produce a fast mode of operation.
- Requires changes in the wiring among the various components if the design has to be modified or changed.

Microprogrammed Control

- The control information is stored in a control memory.
- The control memory is programmed to initiate the required sequence of microoperations.
- Required changes or modifications can be done by updating the microprogram in control memory.

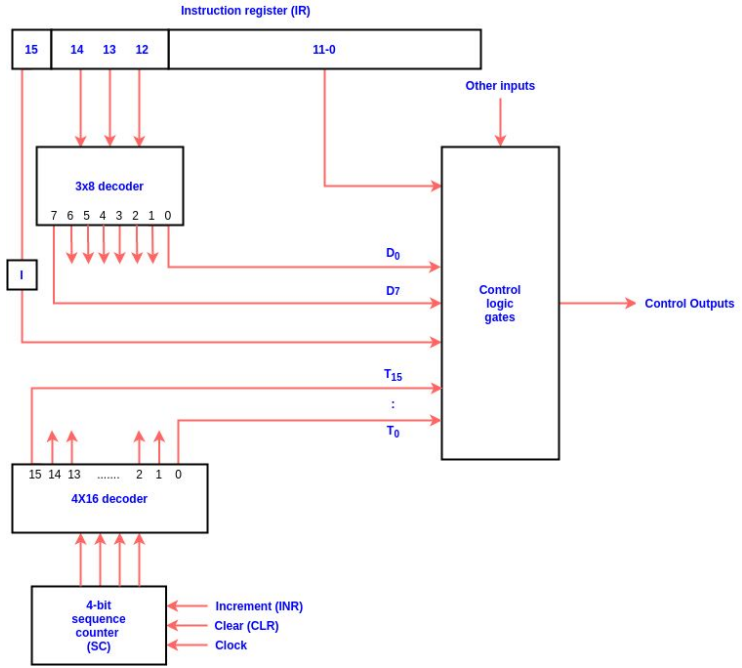
Hardwired control unit

- Consists of two decoders, a sequence counter, and a number of control logic gates.
- An instruction read from memory is placed in the instruction register(IR).
- The IR is divided into three parts:
  - I bit, opcode, and Address bits.
- Op-code in 12-14 bits are decoded with a 3x8 decoder.
- 8 outputs of the decoder are designated by the symbols D0 through D7.
- Bit 15 is transferred to a flip-flop I.
- Bits 0-11 are applied to the control logic gates.



Hardwired control unit

- The 4-bit sequence counter can count in binary from 0-15.
- The outputs of the counter are decoded into 16 timing signals T_0-T_{15}.
- The sequence counter SC can be incremented or cleared synchronously.
- Mostly,SC is incremented to provide the sequence of timing signals(T_1,T2,...,T15)
- Once in a while, the counter is cleared to 0, causing the next active timing signal to be T_0.
- Eg: Suppose,at time T4, SC is cleared to 0 if decoder output D3 is active. D_3T_4: SC ⟵ 0.



Hardwired control unit

- The SC responds to the positive transition of the clock.
- Initially, the CLR input of SC is active.
- Hence it clears SC to 0, giving the timing signal T_0 out of the decoder.
- T0 is active during one clock cycle and will trigger only those registers whose control inputs are connected to timing signal T0.
- SC is incremented with every positive clock transition, unless its CLR input is active.
- This produces the sequence of timing signals T_0, T_1, T_2, T_3, T_4 up to T_{15} and back to T_0.