Computer System Architecture

DR. Howida Youssry

BASIC COMPUTER ORGANIZATION AND DESIGN

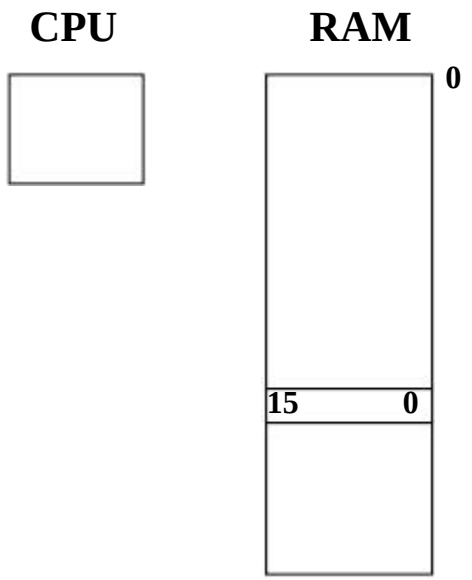
- Instruction Codes
- Computer Registers
- Computer Instructions
- Timing and Control
- Instruction Cycle
- Memory Reference Instructions
- Input-Output and Interrupt
- Complete Computer Description
- Design of Basic Computer
- Design of Accumulator Logic

INTRODUCTION

- **Every different processor type has its own design (different registers, buses, microoperations, machine instructions, etc)**
- **Modern processor is a very complex device**
- **It contains**
  - **Many registers**
  - **Multiple arithmetic units, for both integer and floating point calculations**
  - **The ability to pipeline several consecutive instructions to speed execution**
  - **Etc.**
- **However, to understand how processors work, we will start with a simplified processor model**
- **This is similar to what real processors were like ~25 years ago**
- **M. Morris Mano introduces a simple processor model he calls the *Basic Computer***
- **We will use this to introduce processor organization and the relationship of the RTL model to the higher level computer processor**

THE BASIC COMPUTER

- **The Basic Computer has two components, a processor and memory**
- **The memory has 4096 words in it**
  - **4096 = 2^{12}, so it takes 12 bits to select a word in memory**
- **Each word is 16 bits long**



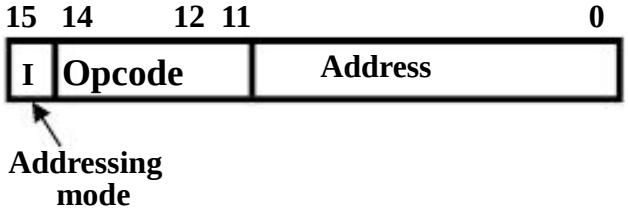
INSTRUCTIONS

- **Program**
  - **A sequence of (machine) instructions**
- **(Machine) Instruction**
  - **A group of bits that tell the computer to *perform a specific operation* (a sequence of micro-operation)**
- **The instructions of a program, along with any needed data are stored in memory**
- **The CPU reads the next instruction from memory**
- **It is placed in an *Instruction Register* (IR)**
- **Control circuitry in control unit then translates the instruction into the sequence of microoperations necessary to implement it**

INSTRUCTION FORMAT

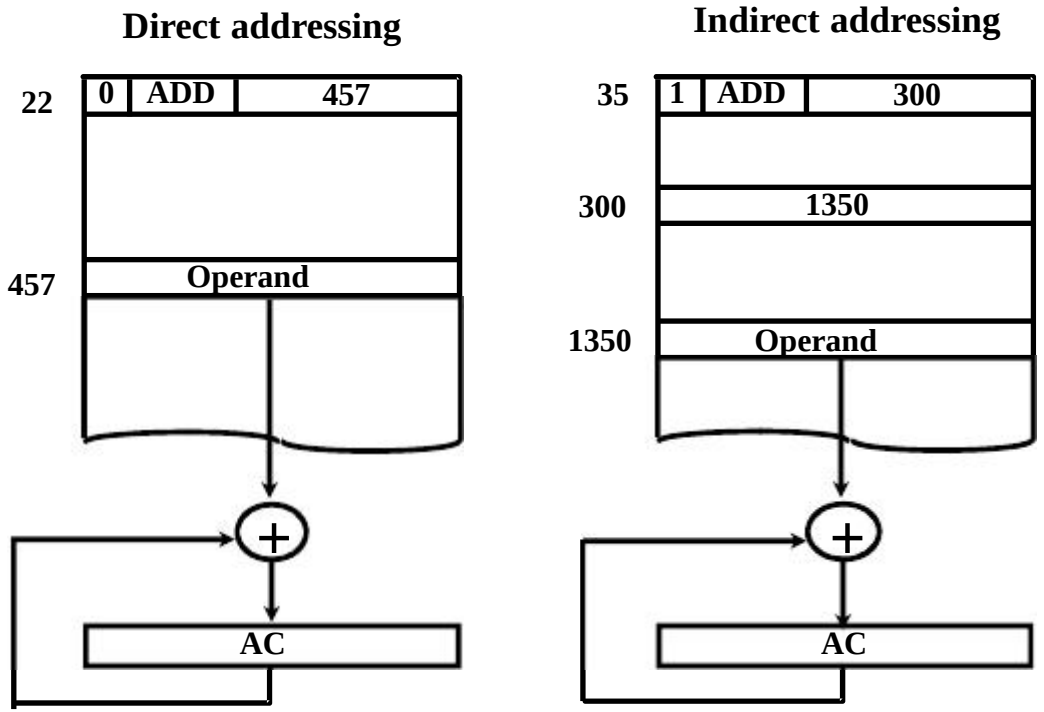
- **A computer instruction is often divided into two parts**
  - **An *opcode* (Operation Code) that specifies the operation for that instruction**
  - **An *address* that specifies the registers and/or locations in memory to use for that operation**
- **In the Basic Computer, since the memory contains 4096 (= 2_{12}) words, we needs 12 bit to specify which memory address this instruction will use**
- **In the Basic Computer, bit 15 of the instruction specifies the *addressing mode* (0: direct addressing, 1: indirect addressing)**
- **Since the memory words, and hence the instructions, are 16 bits long, that leaves 3 bits for the instruction's opcode**

Instruction Format



ADDRESSING MODES

- **The address field of an instruction can represent either**
  - **Direct address: the address in memory of the data to use (the address of the operand), or**
  - **Indirect address: the address in memory of the address in memory of the data to use**



- **Effective Address (EA)**
  - **The address, that can be directly used without modification to access an operand for a computation-type instruction, or as the target address for a branch-type instruction**

PROCESSOR REGISTERS

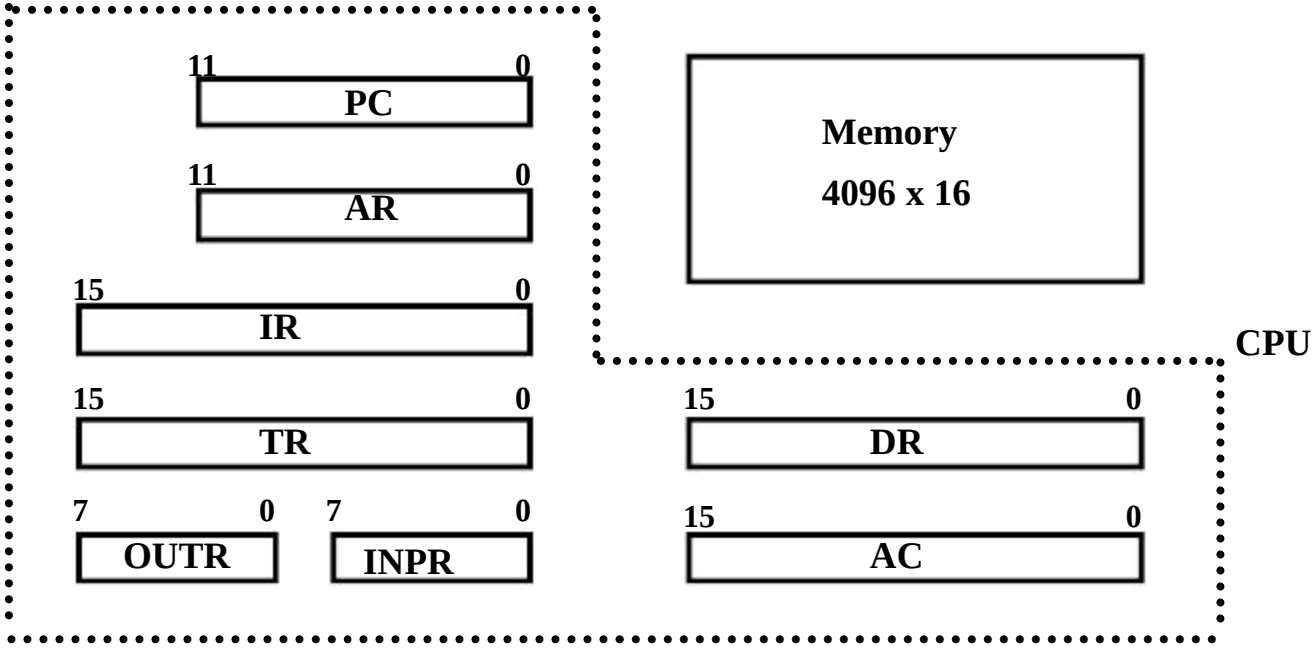
- **A processor has many registers to hold instructions, addresses, data, etc**
- **The processor has a register, the *Program Counter* (PC) that holds the memory address of the next instruction to get**
  - **Since the memory in the Basic Computer only has 4096 locations, the PC only needs 12 bits**
- **In a direct or indirect addressing, the processor needs to keep track of what locations in memory it is addressing: The *Address Register* (AR) is used for this**
  - **The AR is a 12 bit register in the Basic Computer**
- **When an operand is found, using either direct or indirect addressing, it is placed in the *Data Register* (DR). The processor then uses this value as data for its operation**
- **The Basic Computer has a single *general purpose register* – the *Accumulator* (AC)**

PROCESSOR REGISTERS

- **The significance of a general purpose register is that it can be referred to in instructions**
  - **e.g. load AC with the contents of a specific memory location; store the contents of AC into a specified memory location**
- **Often a processor will need a scratch register to store intermediate results or other temporary data; in the Basic Computer this is the *Temporary Register* (TR)**
- **The Basic Computer uses a very simple model of input/output (I/O) operations**
  - **Input devices are considered to send 8 bits of character data to the processor**
  - **The processor can send 8 bits of character data to output devices**
- **The *Input Register* (INPR) holds an 8 bit character gotten from an input device**
- **The *Output Register* (OUTR) holds an 8 bit character to be send to an output device**

BASIC COMPUTER REGISTERS

Registers in the Basic Computer

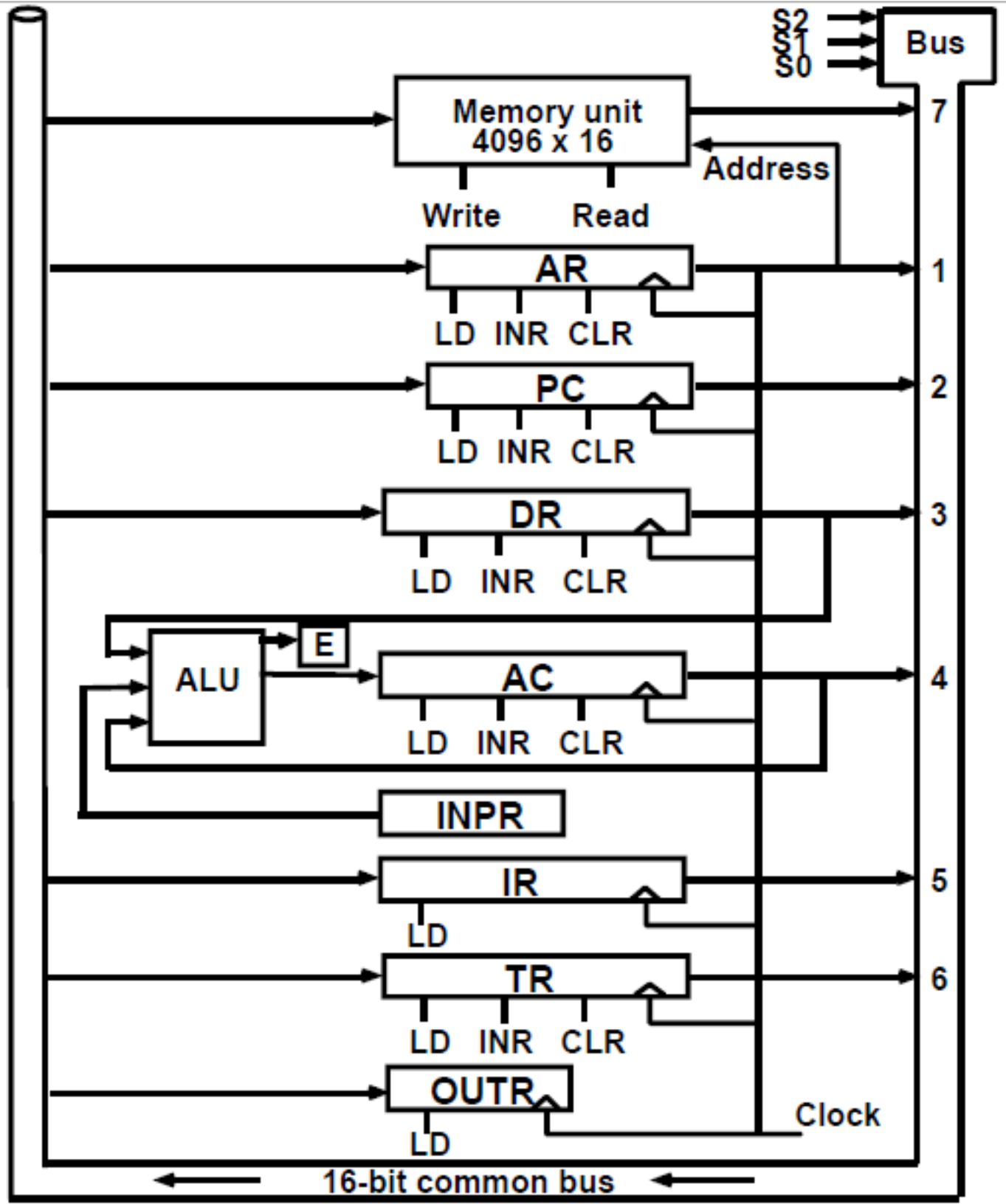


List of BC Registers

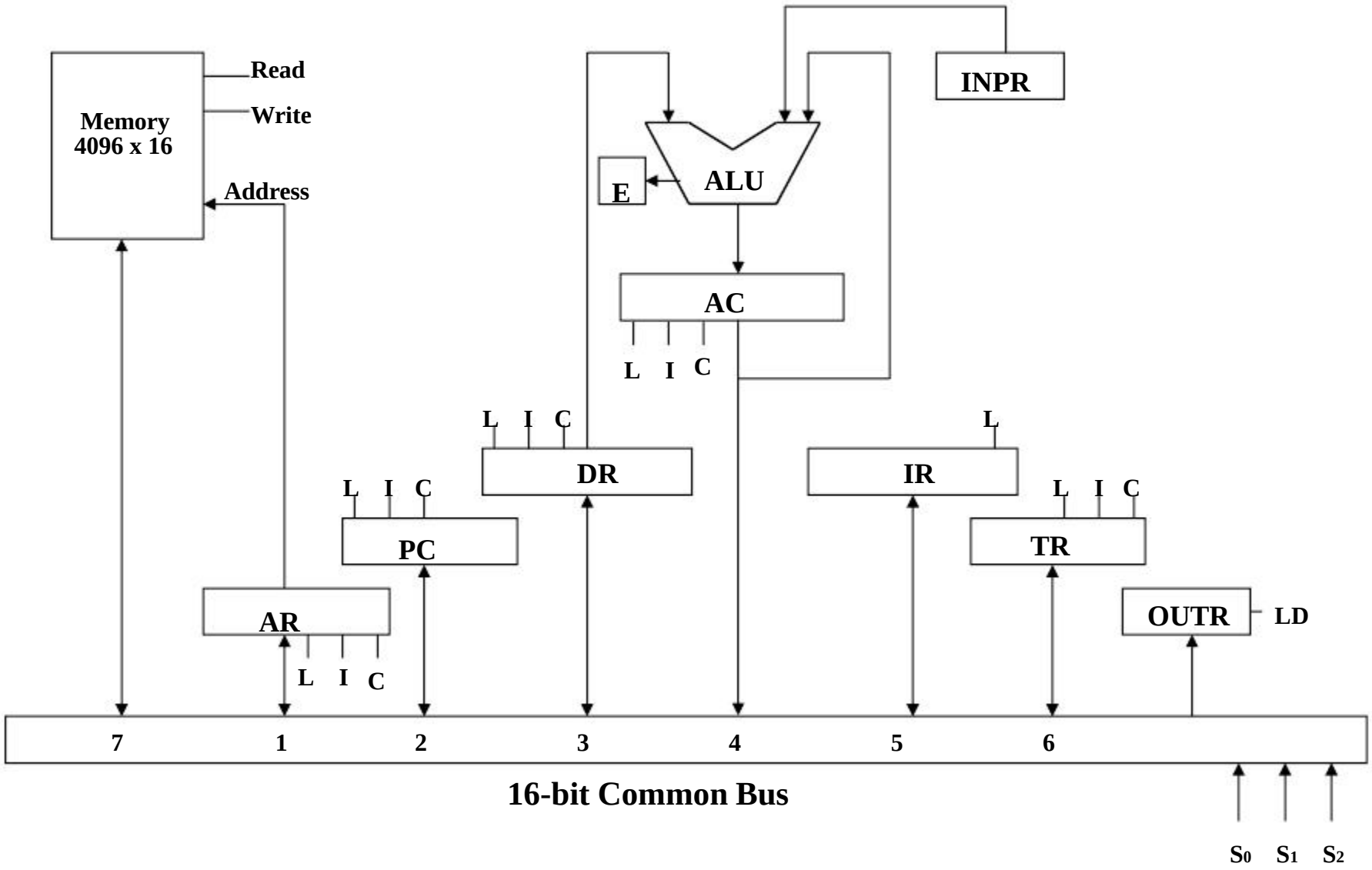
DR	16	Data Register	Holds memory operand
AR	12	Address Register	Holds address for memory
AC	16	Accumulator	Processor register
IR	16	Instruction Register	Holds instruction code
PC	12	Program Counter	Holds address of instruction
TR	16	Temporary Register	Holds temporary data
INPR	8	Input Register	Holds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

COMMON BUS SYSTEM

- **The registers in the Basic Computer are connected using a bus**
- **This gives a savings in circuitry over complete connections between registers**


S2
S1
S0
Bus
Memory unit
4096 x 16
Address
Write
Read
7
AR
LD INR CLR
1
PC
LD INR CLR
2
DR
LD INR CLR
3
ALU
E
AC
LD INR CLR
4
INPR
IR
LD
5
TR
LD INR CLR
6
OUTR
LD
Clock
16-bit common bus

COMMON BUS SYSTEM



COMMON BUS SYSTEM

- **Three control lines, S_2, S_1, and S_0 control which register the bus selects as its input**

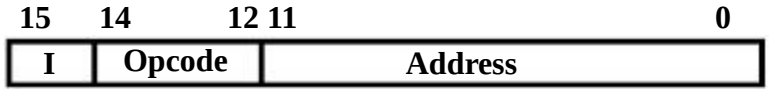
S_2	S_1	S_0	Register
0	0	0	x
0	0	1	AR
0	1	0	PC
0	1	1	DR
1	0	0	AC
1	0	1	IR
1	1	0	TR
1	1	1	Memory

- **Either one of the registers will have its load signal activated, or the memory will have its read signal activated**
  - **Will determine where the data from the bus gets loaded**
- **The 12-bit registers, AR and PC, have 0's loaded onto the bus in the high order 4 bit positions**
- **When the 8-bit register OUTR is loaded from the bus, the data comes from the low order 8 bits on the bus**

BASIC COMPUTER INSTRUCTIONS

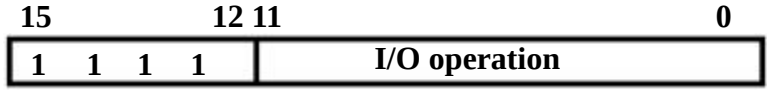
- **Basic Computer Instruction Format**

Memory-Reference Instructions **(OP-code = 000 ~ 110)**



Register-Reference Instructions **(OP-code = 111, I = 0)**

Input-Output Instructions **(OP-code =111, I = 1)**



BASIC COMPUTER INSTRUCTIONS

Symbol	Hex Code I=0	Hex Code I=1	Description
AND	0xxx	8xxx	AND memory word to AC
ADD	1xxx	9xxx	Add memory word to AC
LDA	2xxx	Axxx	Load AC from memory
STA	3xxx	Bxxx	Store content of AC into 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 instr. if AC is positive
SNA	7008		Skip next instr. if AC is negative
SZA	7004		Skip next instr. if AC is zero
SZE	7002		Skip next instr. if E is zero
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

A computer should have a set of instructions so that the user can construct machine language programs to evaluate any function that is known to be computable.

- **Instruction Types**

  Functional Instructions
  - **Arithmetic, logic, and shift instructions**
  - **ADD, CMA, INC, CIR, CIL, AND, CLA**

  Transfer Instructions
  - **Data transfers between the main memory and the processor registers**
  - **LDA, STA**

  Control Instructions
  - **Program sequencing and control**
  - **BUN, BSA, ISZ**

  Input/Output Instructions
  - **Input and output**
  - **INP, OUT**