Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique

uOttawa
L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

CEG2136: Computer Architecture I
CEG2536: Architecture des Ordinateurs I

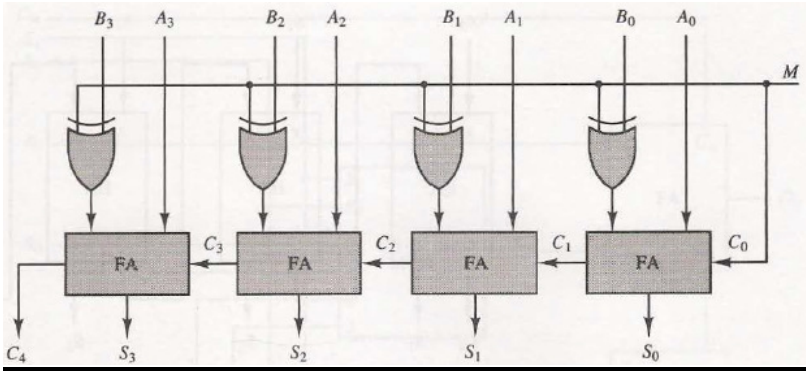
FINAL EXAMINATION

Profs. Voicu Z. Groza and Naim M. Rahmani

- Closed book exam. All electronic devices including calculators are not allowed.
- If in doubt how to interpret a question, make an assumption and elaborate your solution based on this hypothesis. Explain all your assumptions and well define the symbols used.
- If you finish 10 minutes or less before the due time, remain seated until the end of the exam.

Question

Consider the following logic diagram:



a) What is the C_4, S_3, S_2, S_1, S_0 value if:

(1) Register A = 1101, Register B = 0110, and M=0

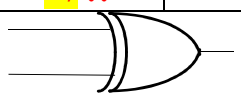
(2) Register A = 1101, Register B = 0110, and M=1

Write your calculations and results in the last two columns of the following table

b) Given that registers A, B and S contain signed numbers in 2's complement representation, convert the binary numbers A_3 A_2 A_1 A_0, B_3 B_2 B_1 B_0 and S_3 S_2 S_1 S_0 to decimal, and write your results and calculations in the following table:

A_3 A_2 A_1 A_0 (binary/ decimal)	$B_3B_2B_1B_0$ (binary/ decimal)	M	C_4	a) S_3 S_2 S_1 S_0 (binary)	b) S_3 S_2 S_1 S_0 (decimal)	c) Are the results provided by the logic circuit correct?, i.e., are they in accord with the results you obtain if you do the same operations yourself
A=1101 = -|A| |A|=-1101= 2's compl(1101) |A|=0011 = 3 A = -3	0110 = + 6	0	1	Carry: $c_4c_3c_2c_1c_0$ 1 1 0 0 0 1101 +0110 =(1)0011	S = A + B S = - 3 + 6 =0011=+3	CORRECT!
1101	0110 = + 6	1	1	Carry : 1 0 011 1101 +1001 (1)0111	S = A – B S = -3 – 6 = 0111= +7 !?	A -B = -3 – 6 = -9 => Overflow, since the result should be in the domain [-8, +7)

d) Expand your adder with a circuit that can signalize overflow



Question

Consider the computer of Lab 4, the architecture of which is described in Figure 1 and Tables 1, 2, 3, 4, and 5. The instruction type is determined by the two most significant bits of the 8-bit register IR, as follows:

X_0 = IR(7)' IR(6)' denotes a memory-reference instruction (MRI) in direct addressing mode;

X1 = IR(7)' IR(6) denotes a register-reference instruction (RRI); and

X_2 = IR(7) IR(6)' denotes a memory-reference instruction (MRI) in indirect addressing mode.

Assume that all registers are equipped with 3 control bits for loading the register (LD), increment by 1 (INC), and reset to zero (CLR).

Find the list of all micro-operations which change the value of register AC and derive the logic equations

T7 Y0 : AC ← AC + DR ,
T7 Y1 : AC ← DR
T7 Y5 : AC ← AC ∧ DR
T7 Y6 : AC ← AC – DR

T3 X1 IR(0) : AC ← 0
T3 X1 IR(1) : AC ← AC
T3 X1 IR(2) : AC ← ashl AC
T3 X1 IR(3) : AC ← ashr AC
T3 X1 IR(4) : AC ← AC + 1

LD_{AC} = T7 (Y0 + Y1 + Y5 + Y6) +
T3[(X1 IR(1) + X1 IR(2) + X1 IR(3)]

INC_{AC} = T3 X1 IR(0)

CLR_{AC} = T3 X1 IR(4)

RRI

Symbol	Notation (RTL)
CLA	T3 X1 IR(0): AC ← 0
CMA	T3 X1 IR(1): AC ← AC
ASL	T3 X1 IR(2): AC ← ashl AC
ASR	T3 X1 IR(3): AC ← ashr AC
INC	T3 X1 IR(4): AC ← AC + 1
HLT	T3 X1 IR(5) : S ← 1

MRI

	Notation (RTL)
Y0 = IR(6)' IR(1) IR(0)'	T6 Y0:DR ← M [AR] T7 Y0:AC← AC + DR SC← 0
Y1 = IR(6)' IR(2)	T6 Y1:DR ← M [AR] T7 Y1:AC← DR, SC← 0
Y2 = IR(6)' IR(3)	T6: T7 Y2:M [AR] ← AC, SC←0
Y3 = IR(6)' IR(4)	T6 Y3:PC← AR , SC← 0
Y4 = IR(6)' IR(5)	T6 Y4:DR ← M [AR] T7 Y4:DR ← DR + 1 T8 Y4:M [AR] ← DR T_9 Y_4:if (DR= 0)S' then (PC← PC+ 1) $T_{10}Y_4$:if (DR = 0)S' then(PC← PC+ 1) T10 Y4:SC← 0
Y5 = IR(6)'IR(1)'IR(0)	T6 Y5:DR ← M [AR] T7 Y5:AC← AC ∧ DR SC← 0
Y6 = IR(6)' IR(1)'IR(0)	T6 Y6:DR ← M [AR] T7 Y6:AC← AC − DR SC← 0

Question

Two memory-reference instructions in the Basic Computer are to be changed to the instructions specified in the following table.

Symbol	Opcode	Symbolic designation	Description in words
ADM	001	$M[EA] \leftarrow AC + M[EA]$	Add AC to memory
BNA	011	If $(AC < 0)$ then $(PC \leftarrow EA)$	Branch if AC is negative

In the following table, write down the necessary micro-operations (in RTL) to perform the execution phase of each instruction. Note that the execution phase of such type of instructions starts at T_4. See figures 2 and 3 from Annex. **Note** that the value in AC should not be changed by the execution of any instruction unless the instruction specifies a change in its content. To this extent you can use TR to store the content of AC temporary.

Question

Column (3) of the following table shows the content of a memory segment of the Basic Computer that stores a machine language program and its operands.

1. Convert from binary to hexadecimal both the address and the memory contents (columns (1) and (3)) and write your results in columns (2) and (4), respectively.
2. Use Table 6 from Annex to convert the machine language program (stored in the first six memory locations) to symbolic operation codes (assembly language). Use Table 6 from Annex to convert

   the machine language program (stored in the first six memory locations) to symbolic operation codes (assembly language). Write the assembly language instructions in the corresponding cells of column (5) of the above table. Use hexadecimal representation for the address field of the instructions.
3. Fill out the last column (6) of the table with the content of the accumulator after the execution of each instruction.

Symbol	RTL
ADM	D_1T_4: $TR \leftarrow AC$ D_1T_5: $DR \leftarrow M[EA]$ D_1T_6: $AC \leftarrow AC + DR$ D_1T_7: $M[EA] \leftarrow AC$ D_1T_8: $DR \leftarrow TR$ D_1T_9: $AC \leftarrow DR, SC \leftarrow 0$
BNA	D_3T_4: If $(AC(15) = 1)$ then $(PC \leftarrow AR)$, $SC \leftarrow 0$

Memory Address		Memory Content		Assembly /operands in HEX	**AC** after execution of instr.
Binary (1)	Hex (2)	Binary (3)	Hex (4)	(5)	(6)
1000 0001 0111	817	0111 1000 0000 0000	7800	CLA	0000
1000 0001 1000	818	0010 1000 0001 1101	281D	LDA 81D	0121
1000 0001 1001	819	0001 1000 0001 1110	181E	ADD 81E	0130
1000 0001 1010	81A	1000 1000 0001 1111	881F	AND 81F I	0100
1000 0001 1011	81B	0011 1000 0001 1101	381D	STA 81D	0100
1000 0001 1100	81C	0111 0000 0000 0001	7001	HLT	0100
1000 0001 1101	81D	0000 0001 0010 0001	0121		
1000 0001 1110	81E	0000 0000 0000 1111	000F		
1000 0001 1111	81F	0000 1000 0010 0000	0820		
1000 0010 0000	820	0000 0001 1100 0100	01C4		

Symbol	Hex code	Symbol	Hex code
AND	0 or 8	CIR	7080
ADD	1 or 9	CIL	7040
LDA	2 or A	INC	7020
STA	3 or B	SPA	7010
BUN	4 or C	SNA	7008
BSA	5 or D	SZA	7004
ISZ	6 or B	SZE	7002
CLA	7800	HLT	7001
CLE	7400	INP	F800
CMA	7200	OUT	F400
CME	7100	SKI	F200

Question

a) Write a subroutine (MUL) in assembly language that multiplies two positive numbers by a repeated addition method. For example, to multiply 6 x 4, the subroutine evaluates the product by adding the multiplicand (6) four times (multiplier), i.e., 6+6+6+6.
Write your program using instructions of the Basic Computer given in the Table 6 from the Annex. Your subroutine should be stored in the memory of your Basic Computer beginning with address **A00**.

Before invoking your subroutine MUL, the calling program places:

- the multiplicand (6 in the above example) in memory location at address **9FF**;
- the multiplier (4 in the above example) in the accumulator.

The result (the product of the two numbers) is passed by subroutine to the calling program through the accumulator.
NOTE: You are allowed to use different ways to pass parameters, but full explanations should be provided.

Solution:

```
        ORG HEX 9FF
MLP,    0000
MUL,    0000
        CMA
        INC          /initialize CTR to
        STA CTR /"negative multiplier"
        CLA
LOP,    ADD MLP
        ISZ CTR
        BUN LOP
        BUN MUL I
CTR,    0000
```

b) This part is independent of part (a).
The coordinates of two 2-dimensional vectors (a_x, a_y, b_x, b_y) are stored at consecutive addresses starting at F01. Write an assembly language program that calculates the scalar (dot) product of two 2-dimensional vectors, assuming that all their coordinates are positive numbers:

$$p = a_x b_x + a_y b_y$$

Your program is stored in memory beginning with location at address **100,** and it has to place the product "p" at address **F00**.

To calculate partial products ($a_x b_x$ and $a_y b_y$), you should call subroutine MUL. Parameters have to be passed as follows:

- memory location **9FF** is used to pass a factor, say a_x;
- the other factor (i.e., b_x) is passed to the subroutine through the accumulator;
- the product $a_x b_x$ is passed back to your program by subroutine MUL through the accumulator.

Solution:

```
        ORG HEX 100
        CLA                 /to initialise P = 0
        STA P
        LDA AX  /prepare ax in MLP
        STA MLP
        LDA BX  /prepare bx in AC
        BSA MUL     / call MUL to get ax bx
        ADD P       / P <- P + ax bx = 0+ ax bx
        STA P
        LDA AY  /prepare ay in MLP
        STA MUL
        LDA BY      /prepare by in AC
        BSA MUL     / call MUL to get ay by
        ADD P   / P <- P + ay by = ax bx + ay by
        STA P
        HLT
        ORG HEX F00
P
AX
AY
BX
BY
```

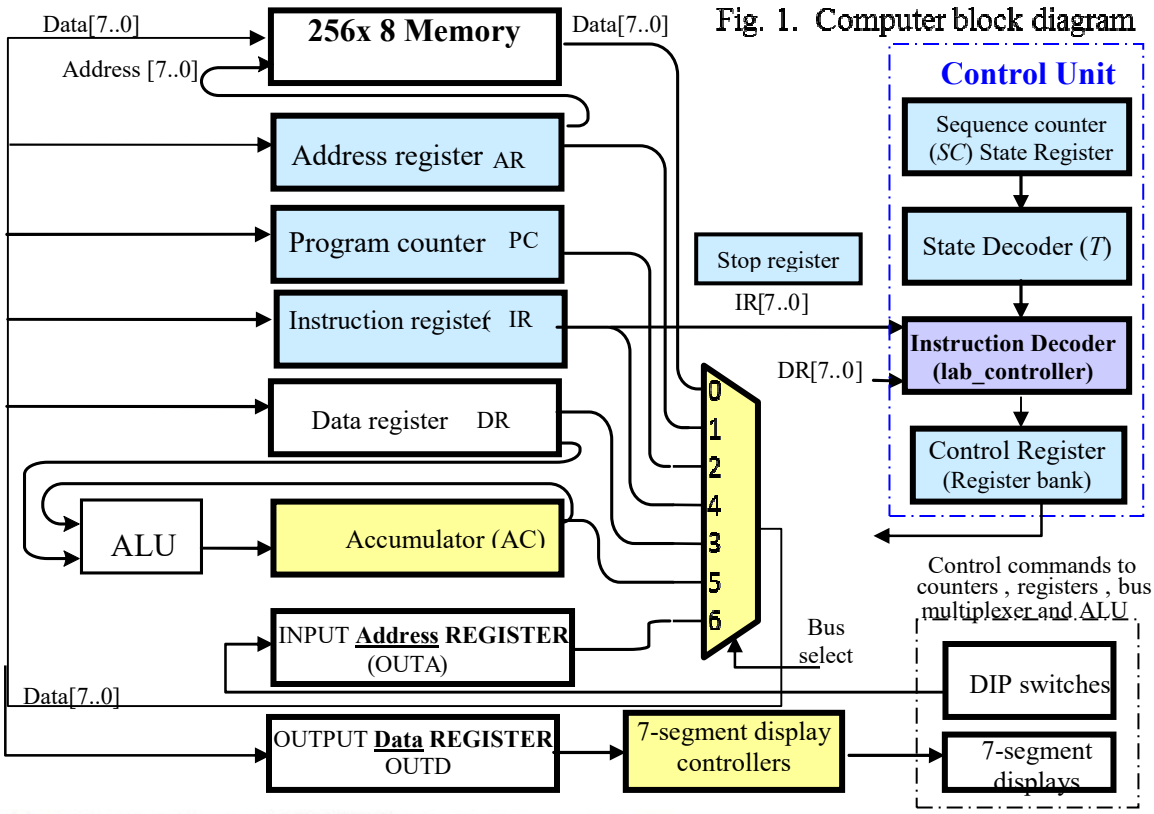


Fig. 1. Computer block diagram

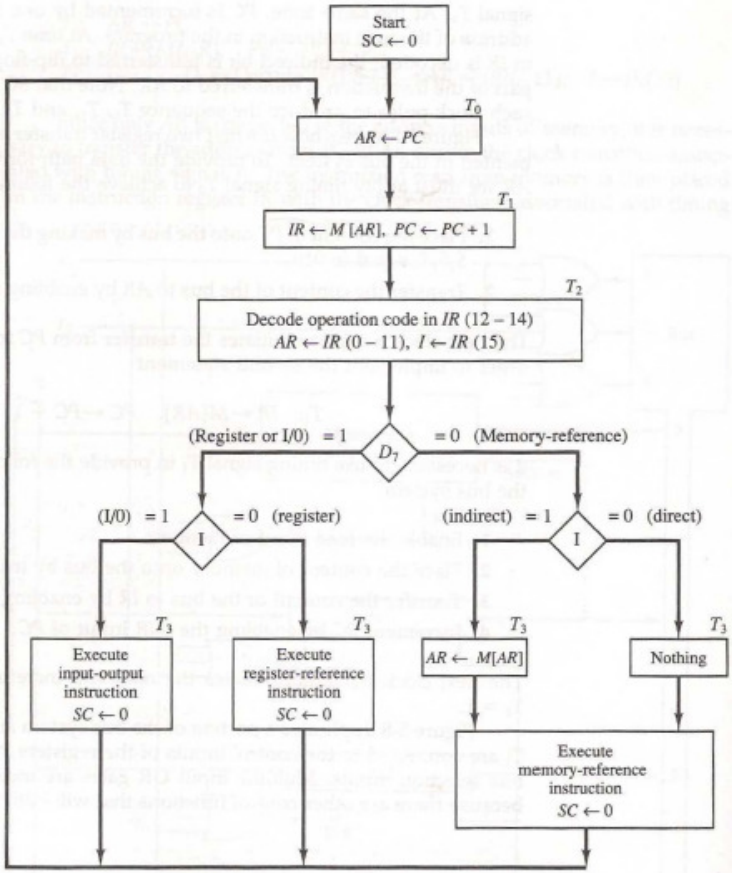


Fig. 2. Flowchart for instruction cycle

Table 1: Starting micro-operations in the instruction cycle of the 8-bit mini computer

Instant	Description	Notation (RTL)
T_0	Load the program counter PC in AR, and increment PC	T_0 : AR ← PC $T_0 S'$: PC ← PC + 1
T_1	Read instruction from memory and put it in the instruction register IR	T_1 : IR ← M [AR]
T_2	This cycle is not used to allow for the new value of the instruction to propagate in the controller	(nothing)
T_3	If X_1 => it is a *register - reference instruction* that will be executed now	$T_3 X_1$: execute an RRI instruction (Table 4) $T_3 X_1$: SC ← 0
T_3	If X_0 or X_2 , read the memory address of the operand and place it in AR, and increment PC. Recall that $(X_0 + X_2) = IR(6)'$	T_3 IR(6)' : AR ← PC T_3 IR(6)' S' : PC ← PC + 1
T_4	Read memory address into AR	T_4 IR(6)' : AR ← M [AR]
T_5	If *indirect addressing*, read the *operand address* from memory location pointed to by AR	$T_5 X_2$: AR ← M [AR]
T_5	If *direct addressing*, don't do anything, as the operand address is already in AR since T_6	$T_5 X_0$: (nothing)
starting from T_6	Execute the memory-reference instructions (MRI) described in Table 2	(see Table 2)

Table 2: Execution of a memory-reference instruction (MRI) in the 8-bit mini computer

Symbol		Notation (RTL)
ADD	$Y_0 = IR(6)'\ IR(1)\ IR(0)'$	$T_6 Y_0$: DR ← M [AR] $T_7 Y_0$: AC ← AC + DR , SC ← 0
LDA	$Y_1 = IR(6)'\ IR(2)$	$T_6 Y_1$: DR ← M [AR] $T_7 Y_1$: AC ← DR , SC ← 0
STA	$Y_2 = IR(6)'\ IR(3)$	T_6 : (cycle not used to allow for the address bus to stabilize) $T_7 Y_2$: M [AR] ← AC , SC ← 0
BUN	$Y_3 = IR(6)'\ IR(4)$	$T_6 Y_3$: PC ← AR , SC ← 0
ISZ (assuming that the next instruction is a memory-reference instruction, stored at 2 memory locations further down)	$Y_4 = IR(6)'\ IR(5)$	$T_6 Y_4$: DR ← M [AR] $T_7 Y_4$: DR ← DR + 1 $T_8 Y_4$: M [AR] ← DR $T_9 Y_4$: if (DR = 0)S' then (PC ← PC + 1) $T_{10} Y_4$: if (DR = 0)S' then (PC ← PC + 1) $T_{10} Y_4$: SC ← 0
AND	$Y_5 = IR(6)'\ IR(1)'\ IR(0)$	$T_6 Y_5$: DR ← M [AR] $T_7 Y_5$: AC ← AC ∧ DR , SC ← 0
SUB	$Y_6 = IR(6)'\ IR(1)'\ IR(0)$	$T_6 Y_6$: DR ← M [AR] $T_7 Y_6$: AC ← AC − DR , SC ← 0

Table 3:
Function table of the 8-bit ALU

S_2	S_1	S_0	Operation
0	0	0	F = AC + DR
0	0	1	F = AC - DR
0	1	0	F = ashl AC
0	1	1	F = ashr AC
1	0	0	F = AC ∧ DR
1	0	1	F = AC ∨ DR
1	1	0	F = DR (transfer)
1	1	1	F = AC'

Table 4:
Execution of a register-reference instruction (RRI) of the 8-bit mini computer

Symbol	Notation (RTL)
CLA	T_3X_1IR(0) : AC ← 0
CMA	T_3X_1IR(1) : AC ← AC
ASL	T_3X_1IR(2) : AC ← ashl
ASR	T_3X_1IR(3) : AC ← ashr
INC	T_3X_1IR(4) : AC ← AC
HLT	T_3X_1IR(5) : S ← 1

Table 5:
Function table of the 8-bit bus

S_2	S_1	S_0	Register placed on the bus
0	0	0	Memory
0	0	1	AR
0	1	0	PC
0	1	1	DR
1	0	0	IR
1	0	1	AC
1	1	0	OUTA
1	1	1	None

Table 6:
Instructions list of the 16-bit Basic Computer of the textbook

Symbol	Hex code	Description
AND	0 or 8	AND *M* to *AC*
ADD	1 or 9	Add *M* to *AC,* carry to *E*
LDA	2 or A	Load *AC* from *M*
STA	3 or B	Store *AC* in *M*
BUN	4 or C	Branch unconditionally to *m*
BSA	*5* or D	Save return address in *m* and branch to *m* + 1
ISZ	6 or B	Increment *M* and skip if zero
CLA	7800	Clear *AC*
CLE	7400	Clear *E*
CMA	7200	Complement *AC*
CME	7100	Complement *E*
CIR	7080	Circulate right *E* and *AC*
CIL	7040	Circulate left *E* and *AC*
INC	7020	Increment *AC,*
SPA	7010	Skip if *AC* is positive
SNA	7008	Skip if *AC* is negative
SZA	7004	Skip if *AC* is zero
SZE	7002	Skip if *E* is zero
HLT	7001	Halt computer
INP	F800	Input information and clear flag
OUT	F400	Output information and clear flag
SKI	F200	Skip if input flag is on
SKO	F100	Skip if output flag is on
ION	F080	Turn interrupt on
IOF	F040	Turn interrupt off

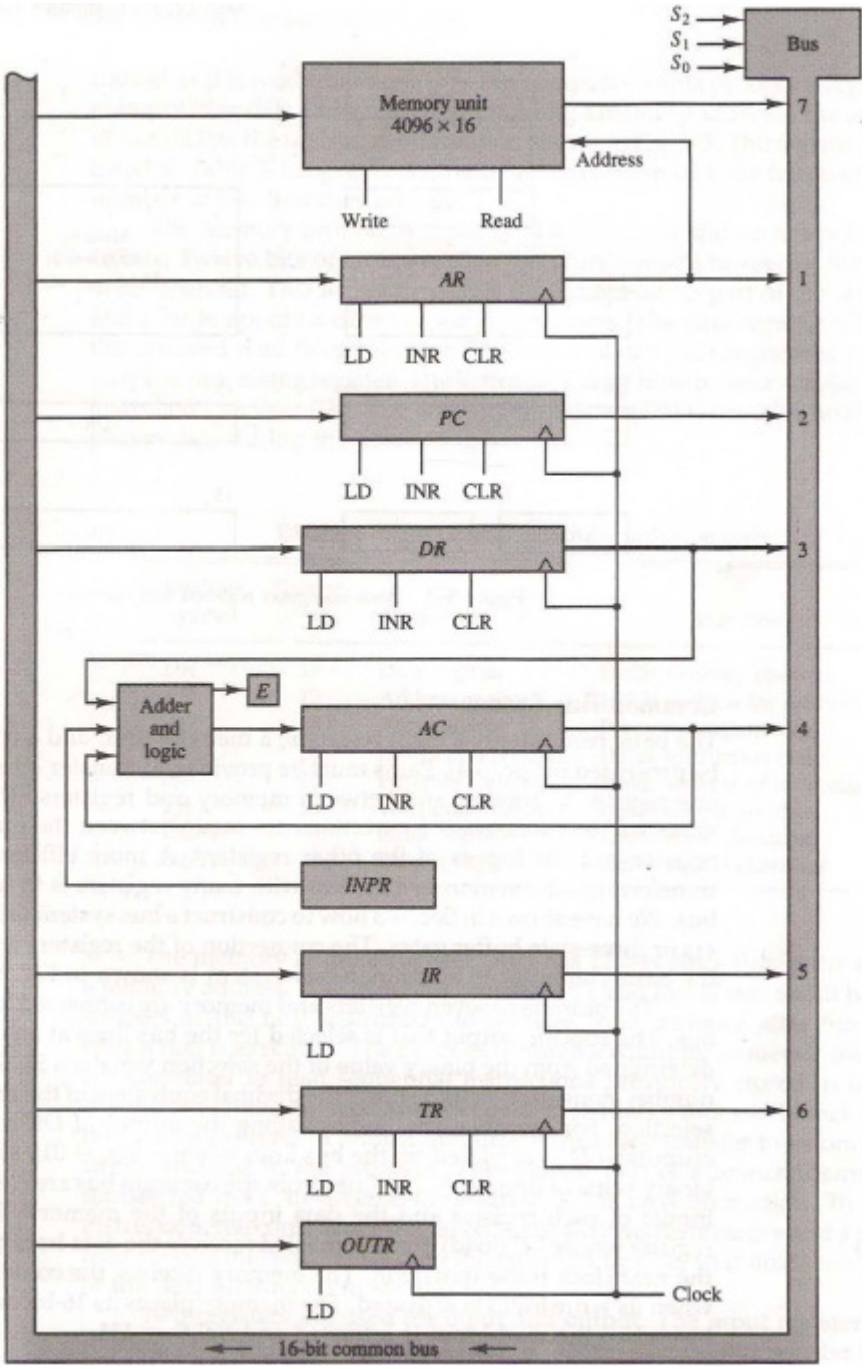


Fig. 3. Basic Computer registers connected to a common bus