Phase 1 Report

PDP-11 design

By

Ahmed Sherif

Mohammed Ibrahim

Ahmed Magdy

Ahmed Ashraf

Instruction Set Architecture

Addressing Modes

Mode	Code
Register Direct	000
Register Indirect	001
Auto Increment Direct	010
Auto Increment Indirect	011
Auto Decrement Direct	100
Auto Decrement Indirect	101
Indexed Direct	110
Indexed Indirect	111

Registers

Register	Code
R_0	000
R_1	001
R_2	010
R_3	011
R_4	100
R_5	101
R_6	110
R_7	111

IR Design

2 Operands

4-bits: Opcode	Src 3-bits: Mode	Src 3-bits: Reg	Dst 3-bits: Mode	Dst 3-bits: Reg

Instruction	Opcode	Operation	NZVC
MOV	0011	Dst ← [Src]	N set if [src] < 0 Z set if [src] = 0 V reset C not affected
ADD	0100	Dst ← [Dst] + [Src]	N set if [src] < 0 Z set if [src] = 0 V is set if arithmetic overflow C set if carry from MSB of result occurs
ADC	0101	Dst ← [Dst] + [Src] + C	N set if result < 0 (MSB = 1) Z set if result = 0 C set if carry from MSB of result occurs
SUB	0110	Dst ← [Dst] – [Src]	N set if result < 0 (MSB = 1) Z set if result = 0 C set if no carry from MSB of result occurs
SBC	0111	Dst ← [Dst] – [Src] – C	N set if result < 0 Z set if result = 0 V set if arithmetic overflow occurs C set if no carry from the MSB of result occurs
AND	1000	Dst ← [Dst] AND [Src]	N set if result MSB = 1 Z set if result = 0 V reset C not affected

OR	1001	Dst ← [Dst] OR [Src]	set Z if output = 0 set N = MSB of output Carry not affected
XOR	1010	Dst ← [Dst] XOR [Src]	set Z if output = 0 set N = MSB of output Carry not affected
CMP	1011	[Dst] – [Src] (Operands are NOT affected)	set N = MSB of output Z set if result = 0 C set if no carry from MSB of result occurs

1 Operand

0000: 1-Operand Flag	4-bits: Opcode	2-bits: Don't Care	Dst 3-bits: Mode	Dst 3-bits: Reg

Instruction	Opcode	Operation	NZVC
INC	0011	Dst ← [Dst] + 1	N set if result] < 0 Z set if result = 0 V is set if overflow [dst] = 077777 C not affected
DEC	0100	Dst ← [Dst] – 1	N set if result] < 0 Z set if result = 0 V is set if overflow [dst] = 100000 C not affected
CLR	0101	Dst ← 0	N reset C reset Z set
INV	0110	Dst ← INV([Dst])	N set if result < 0 (MSB = 1) Z set if result = 0 C reset
LSR	0111	Dst ← 0 || [Dst]$_{15\text{->}1}$	N set if result < 0 Z set if result = 0 V = N XOR C

			C holds the last bit rotated out
ROR	1000	$\text{Dst} \leftarrow [\text{Dst}]_{0} \Vert [\text{Dst}]_{15->1}$	N set if result < 0 Z set if result = 0 V = N XOR C C holds the last bit rotated out
ASR	1001	$\text{Dst} \leftarrow [\text{Dst}]_{15} \Vert [\text{Dst}]_{15->1}$	set Z output = 0 set N= MSB of output C = LSB of input
LSL	1010	$\text{Dst} \leftarrow [\text{Dst}]_{14->0} \Vert 0$	set Z output = 0 set N= MSB of output C =MSB of input
ROL	1011	$\text{Dst} \leftarrow [\text{Dst}]_{14->0} \Vert [\text{Dst}]_{15}$	set Z output = 0 set N= MSB of output C =MSB of input
JSR	1100	SP ← [SP] – 1 [SP] ← [PC] [PC] ← [Address]	Not Affected

Branch

0001: Branch Flag	4-bits: Opcode	8-bits: Offset

Instruction	Opcode	Condition
BR	0011	None
BEQ	0100	Z=1
BNE	0101	Z=0
BLO	0110	C=0

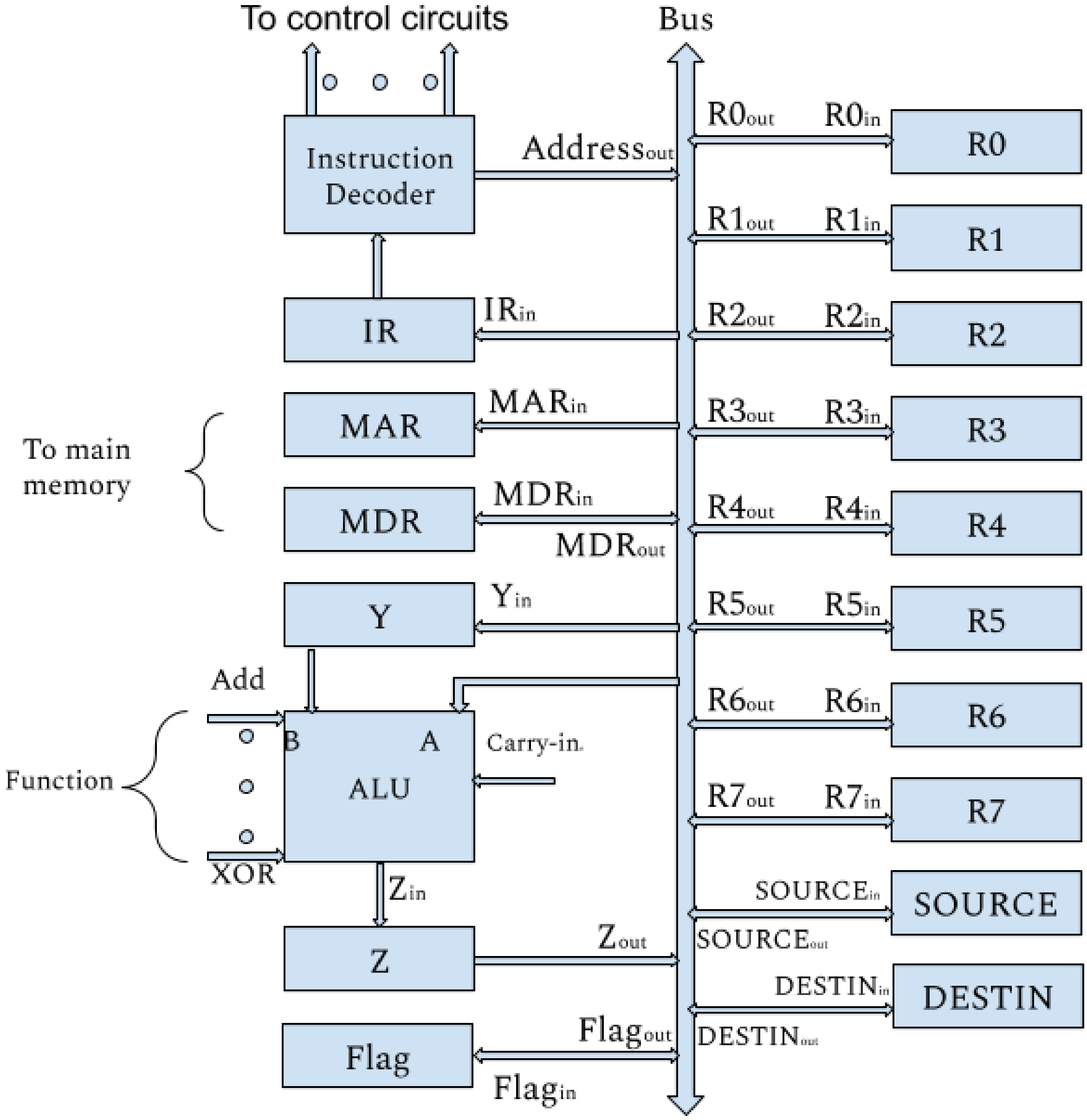
BLS	**0111**	**C=0 OR Z=1**
BHI	**1000**	**C=1**
BHS	**1001**	**C=1 OR Z=1**

No Operand

0010: No Operand Flag	3-bits: Opcode	9-bits: Don’t Care

Instruction	Opcode	Operation
HLT	**000**	**Stop the processor**
NOP	**001**	**No operation is performed** **Continue code**
RESET	**010**	**All devices are reset**
RTS	**011**	[PC] ← [SP] **SP** ← [SP] + **1**
IRET	**100**	**Pop flags from stack** **Return back to PC**

Bus System Schematic



Micro-Instructions

Fetch and Decode

1. **PC_out, MAR_in, Read, Clear_Y, SetCarry, Add, Z_in**
2. **Z_out, PC_in, WMFC**
3. **MDR_out, IR_in**

Fetch Src,DST(READ) :

1) **For Direct :**
    a) **Register :**
        1) **R_out**
    b) **Auto Increment:**

        1) **R_out,MAR_in,Read,Clear_Y,SetCarry,Add,Z_in**
        2) **Z_out,R_in,WMFC**
        3) **MDR_out**
    c) **Auto Decrement:**
        1) **R_out,Clear_Y,SetCarry,Sub,Z_in**
        2) **Z_out,R_in,MAR_in,Read,WMFC**
        3) **MDR_out**
    d) **Indexed:**
        1) **PC_out,MAR_in,Read**
        2) **R_out,Y_in,WMFC**
        3) **MDR_out,Add,Z_in**
        4) **Z_out,MAR_in,Read,WMFC**
        5) **MDR_out**

2) **For Indirect: just add on its direct mode these lines**
    a) **MAR_in,Read,WMFC**
    b) **MDR_out**

⇒ After fetching src put it in the TMP1 register

⇒ After fetching dst put it in the TMP2 register

Save results(Write):

1) **For Register Direct: R_in**
2) **For All Other Direct and Indirect Modes:**
    a) **MDR_in,Write,WMFC,End**

Assume that:

1. **There is a register called the SOURCE/TMP1 to hold the src operand**
2. **There is a register called the DESTIN/TMP2 to hold the dst operand**
3. **For all the ALU operations set the Cin in the ALU to the carry and after performing the operation set the carry to Cout**

Instructions

1. **Fetch Src**
2. **Fetch Dst**
3. **Perform Operation**
4. **Save Result(If needed)**

Instruction	**Operation Performed**
MOV	SOURCE_out, Save Result
ADD	SOURCE_out, Y_in DESTIN_out, ADD, Z_in Z_out, Save Result
ADC	SOURCE_out, Y_in DESTIN_out, ADC, Z_in

	Z_out, Save Result
SUB	SOURCE_out, Y_in DESTIN_out, SUB, Z_in Z_out, Save Result
SBC	SOURCE_out, Y_in DESTIN_out, SBC, Z_in Z_out, Save Result
AND	SOURCE_out, Y_in DESTIN_out, AND, Z_in Z_out, Save Result
OR	SOURCE_out, Y_in DESTIN_out, OR, Z_in Z_out, Save Result
XOR	SOURCE_out, Y_in DESTIN_out, XOR, Z_in Z_out, Save Result
CMP	SOURCE_out, Y_in DESTIN_out, SUB, Z_in Z_out
INC	DESTIN_out, Increment, Z_in Zout, Save Result
DEC	DESTIN_out, Decrement, Z_in Zout, Save Result
CLR	DESTIN_out, CLR, Z_in Zout, Save Result
INV	DESTIN_out, INV, Z_in Zout, Save Result
LSR	DESTIN_out, LSR, Z_in Zout, Save Result
ROR	DESTIN_out, ROR, Z_in Zout, Save Result
ASR	DESTIN_out, ASR, Z_in Zout, Save Result
LSL	DESTIN_out, LSL, Z_in Zout, Save Result
ROL	DESTIN_out, ROL, Z_in Zout, Save Result
Branching	PC_out, Y_in **if CONDITION else END** IR_Addr_out, Y_out, Add, Z_in

	Z_out, PC_in
HLT	End
JSR	PC_out, MAR_in, Read, INC, Z_in Z_out, PC_in SP_out, DEC, Z_in Z_out, SP_in, WMFC MDR_out, TMP1_in SP_out, MAR_in PC_out, MDR_in, WRITE TMP1_out, PC_in, WMFC **END**
RTS	1) SP_out, MAR_in, Read, Clear_Y, SetCarry, Add, Z_in 2) Z_out, SP_in,WMFC 3) MDR_out, PC_in
INTERRUPT	SP_out ,MAR_in,ADC,Z_in Z_out,SP_in FLAG_out,MDR_in,Write SP_out,MAR_in,ADC,Z_in Z_out,SP_in,WMFC PC_out,MDR_in,Write,WMFC IR_ADD_OUT,PC_in
IRET	1. SP_out, MAR_in, Read, Set_Carry, Add,Z_in 2. Z_out, SP_in, WMFC 3. MDR_out, PC_in 4. SP_out, MAR_in, Read, Set_Carry,ADD, Z_in, 5. Z_out, SP_in, WMFC 6. MDR_out, FLAG_in

Memory Access And Clock Cycles:

1 - Addressing Modes

Addressing Modes	**Register Direct**	**Register Indirect**	**Auto-Increment Direct**	**Auto-Increment Indirect**	**Auto-decrement Direct**	**Auto-decrement Indirect**	**Indexed Direct**	**Indexed Indirect**
Memory Access	**0**	**1**	**1**	**2**	**1**	**2**	**2**	**3**
Clock Cycles	**1**	**3**	**3**	**5**	**3**	**5**	**5**	**7**

2 - Instructions

For all Instructions :

2Opernad(Memory Access , Clock Cycles)

Mode	RegisterDirect	RegisterIndirect	Auto-Increment Direct	Auto-Increment Indirect	Auto-Decrement Direct	Auto-Decrement Indirect	Indexed Direct	Indexed Indirect
RegisterDirect	0,2	1,4	1,4	2,6	1,4	2,6	2,6	3,8
RegisterIndirect		2,6	2,6	3,8	2,6	3,8	3,8	4,10
Auto-Increment Direct			2,6	3,8	2,6	3,8	3,8	4,10
Auto-Increment Indirect				4,10	3,8	4,10	4,10	5,12
Auto-Decrement Direct					2,6	3,8	3,8	4,10
Auto-Decrement Indirect						4,10	4,10	5,12
Indexed Direct							4,10	5,12
Indexed Indirect								6,14

Sum of Memory Access = 192

Sum of Clock Cycles = 512

1 Operand

Sum of Clock Cycles = 32

1） 1 Memory Access for Fetch Instruction + Memory Access of Fetching Src(2 operand)as in the Previous Table + Memory Access of Fetching Dst(1&2 Operands) as in the Previous Table + Memory Access needed for operation in the following table.

2） 3 Clock Cycles for Fetching the instruction + Clock Cycles of Fetching Src(2 operand)as in the Previous Table + Clock Cycles of Fetching Dst(1&2 Operands) as in the Previous Table + Clock Cycles needed for operation in the following table.

Note: Write in Table equals 0 if Mode is Register Direct and 1 otherwise.

Instruction	Memory Access	Clock Cycles
MOV	**Write**	**1**
ADD	**Write**	**3**
ADC	**Write**	**3**

SUB	Write	3
SBC	Write	3
AND	Write	3
OR	Write	3
XOR	Write	3
CMP	0	3
INC	Write	2
DEC	Write	2
CLR	Write	2
INV	Write	2
LSR	Write	2
ROR	Write	2
ASR	Write	2
LSL	Write	2
ROL	Write	2
BRANCHING	0	3
HLT	0	1
JSR	2	8
RTS	1	3
INTERRUPT	2	7

IRET	2	6

Average Number of Clock = Sum(clock cycles of all instructions) / Number of instructions.

2 operand + 1 operand + branching + JSR + RTS +IRET + INTERRUPT + HLT / total number of instructions

= (7936 + 648 + 42 + 11 + 6 + 9 + 10 + 4)/(576 + 72 + 7 + 1 + 1 + 1 +1 + 1)

= 8666 / 660 = 13.13 Clk/Instr

-- GENERAL NOTES

1. **The machine is word addressable so we need to pc=pc+2**

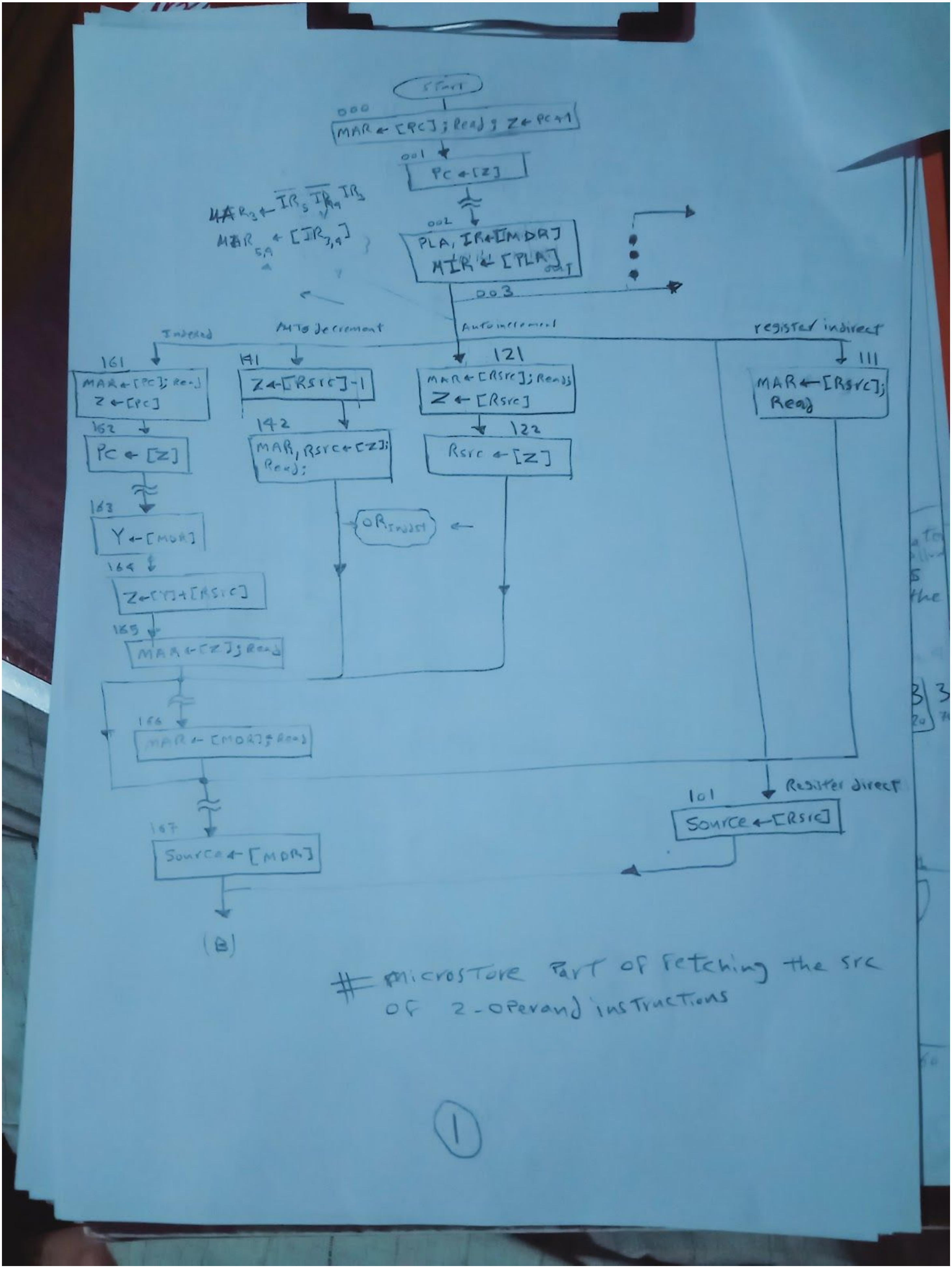
- Control Word Structure

F0		F1		F2		F3		F4	
Address of the next microinstruction		000	No Transfer	000	No Transfer	00	No Transfer	00	No Transfer
		001	Pc-out	001	PC-in	01	MAR-in	01	Y-in
		010	MDR-out	010	IR-in	10	MDR-in	10	SOURCE-in
		011	Z-out	011	Z-in			11	DESTIN-in
		100	Rsrc-out	100	Rsrc-in				
		101	Rdst-out	101	Rds-in				
		110	SOURCE-out						
		111	DESTIN-out						

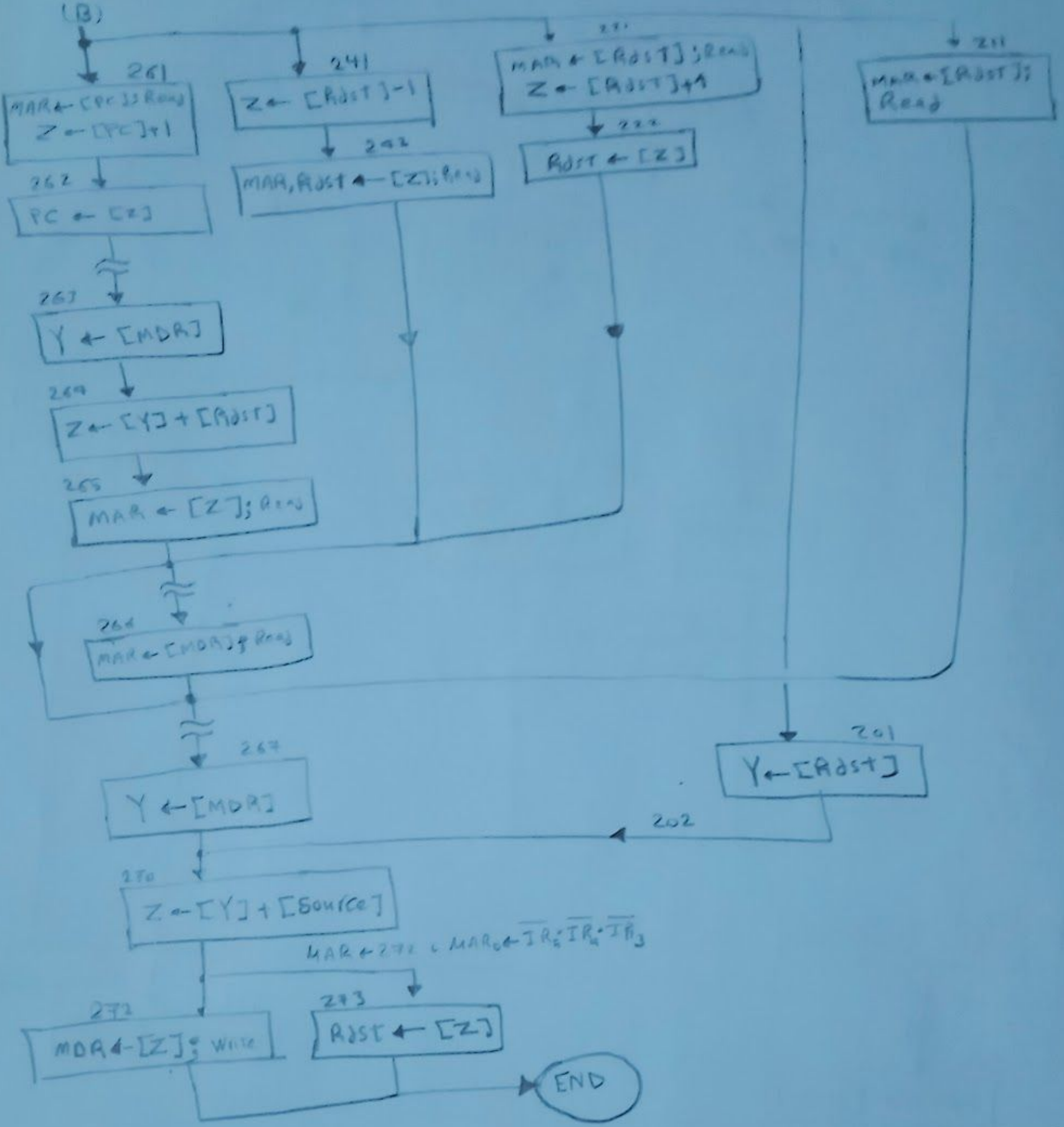
F5		F6		F7		F8		F9	
0100	ADD	00	No Action	0	Clear Y	0	Carry-in =0	0	No Action
0101	ADC	01	Read			1	Carry-in =1	1	WMFC
0110	SUB	10	Write						
0111	SBC								
1000	AND								
1001	OR								
1010	XOR								
1011	CMP								

F10		F11	
000	No Action	0	No Action
001	OR-dst	1	PLA-out
010	OR-indsrc		
011	OR-inddst		

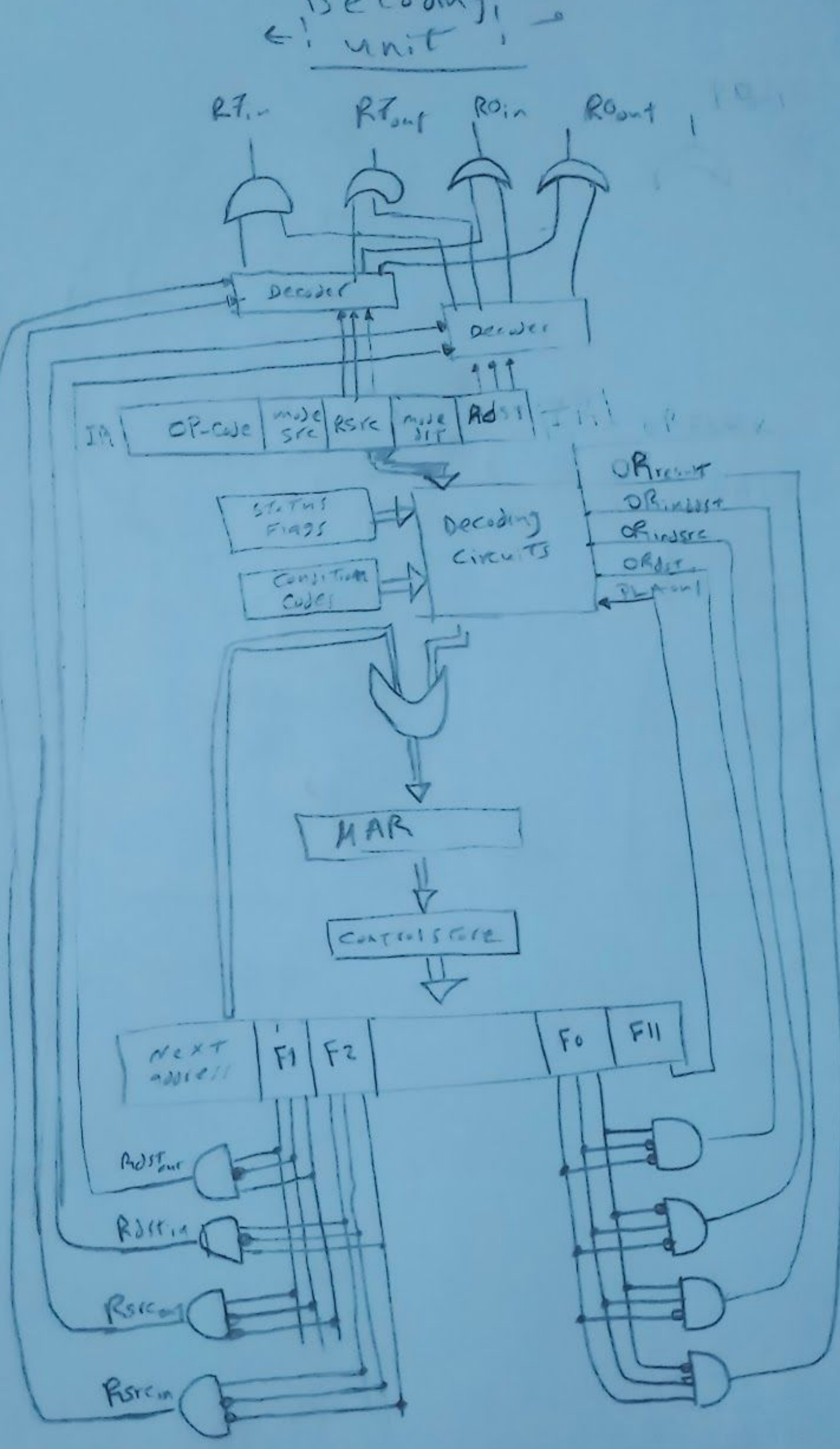
100	OR-result		



Microstore part of fetching the src of 2-operand instructions

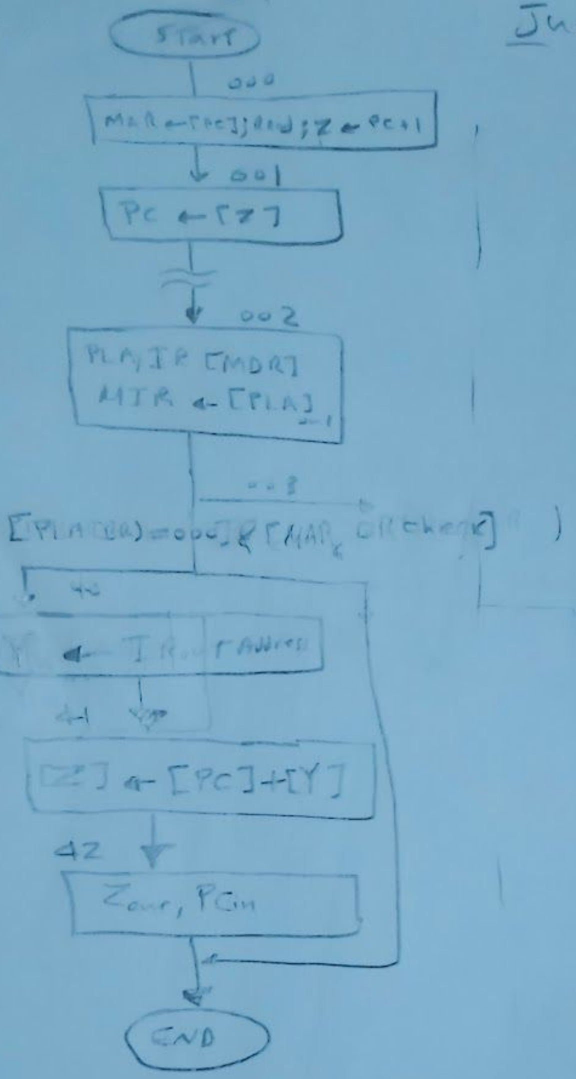


# micro store part of Fetching destination For 2-operand & 1-operand instructions



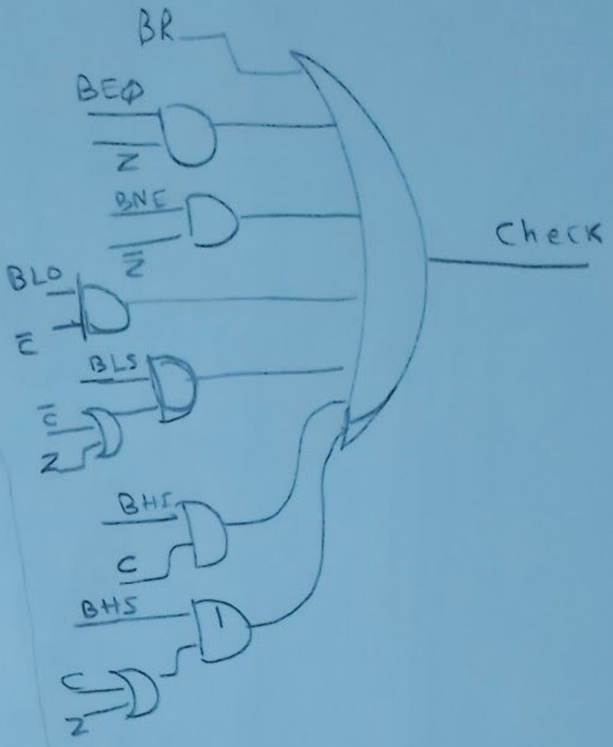
IR Decoding unit

Branch



#microstore Part

Jump instruction



#Decoding Part