FACULTY OF ENGINEERING, CAIRO UNIVERSITY, COMPUTER ENGINEERING DEPARTMENT

Architecture Project

Simple PDP-11

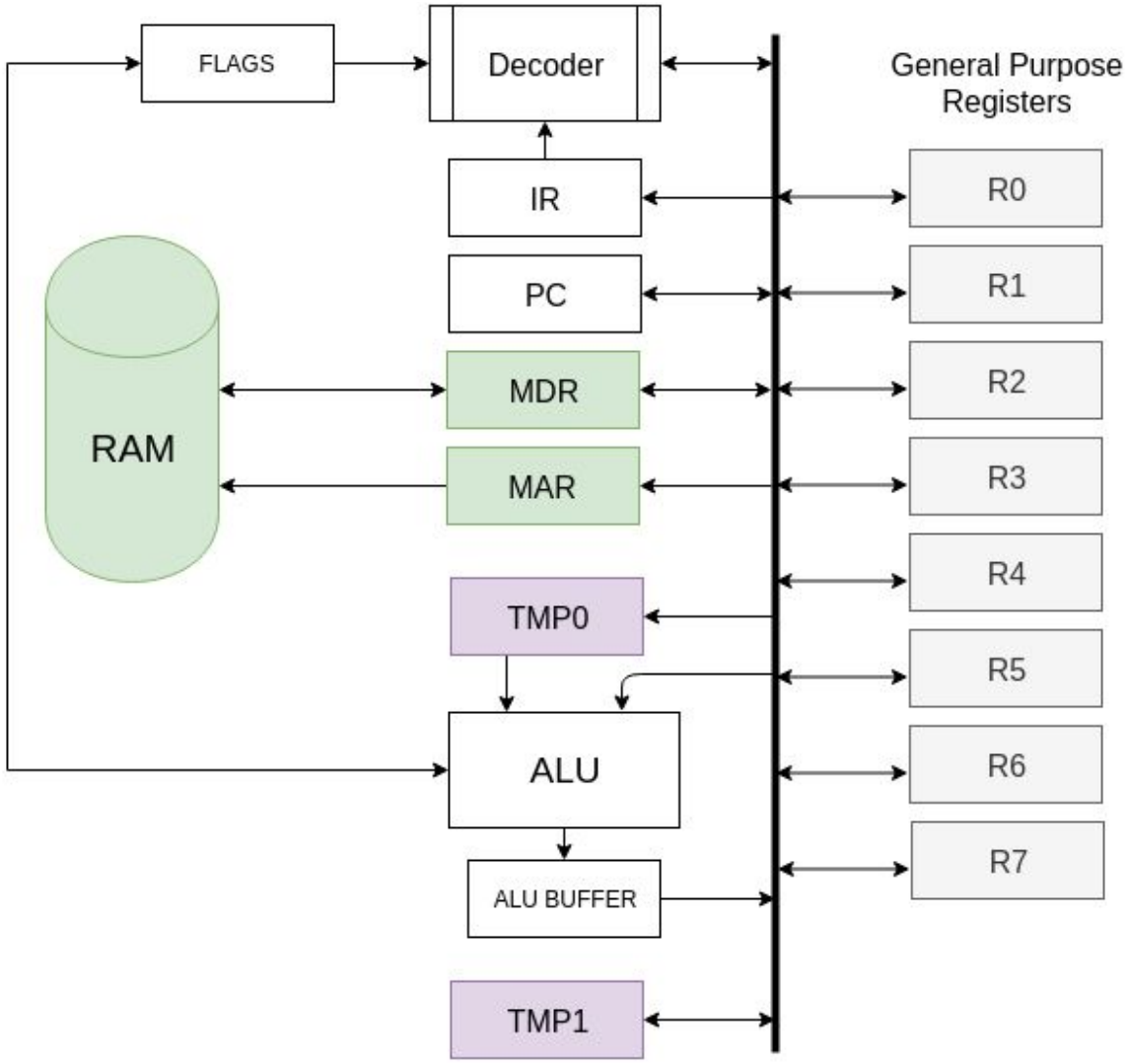
Team 15

Team Members	Sec	Bn
-Evram Youssef	1	9
-Remonda Talaat	1	20
-Mohamed Shawky	2	16
-Mahmoud Othman	2	21

Submitted to: TA.Ali el-Seddiq

Under supervision of: Dr. Hwaida

General Architecture Info:



- Single bus.
- 8 General purpose registers (R0 -> R7).
- 2 TMP registers, ALU buffer and 6-bit FLAGS register.
- 8*1024 16-bit-words RAM.

Two operands instructions:

- ❏ Syntax “Opcode Src, Dst”
- ❏ IR:

4 bits opcode	6 bits src	6 bits dst

Instruction Name	**Op Code (Binary)**	**Operation Performed**	**Condition codes**
Mov	0001	Dst ← [Src]	“N” set if [src] < 0 “Z” set if [src] = 0 “V” reset
ADD	0010	Dst ← [Dst] + [Src]	“N” set if result < 0 “Z” set if result = 0 “V” set if arithmetic overflow occurs “C” set if carry from the MSB of result occurs
ADC	0011	Dst ← [Dst] + [Src] + C	“N” set if result < 0 “Z” set if result = 0 “V” set if arithmetic overflow occurs “C” set if carry from the MSB of result occurs
SUB	0100	Dst ← [Dst] – [Src]	“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

SBC	0101	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	0110	Dst ← [Dst] AND [Src]	“N” set if MSB of result = 0 “Z” set if [src] = 0 “V” reset
OR	0111	Dst ← [Dst] OR [Src]	“N” set if MSB of result = 0 “Z” set if [src] = 0 “V” reset
XNOR	1000	Dst ← [Dst] XNOR [Src]	“N” set if MSB of result = 0 “Z” set if [src] = 0 “V” reset
CMP	1001	[Dst] – [Src] Neither of the operands are affected	“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

One operand instructions:

- ❏ Syntax “Opcode Dst”
- ❏ IR:

8bits opcode	8bits dst

Instruction Name	Op Code (Binary)	Operation Performed	Condition codes
INC	11110000	Dst ← [Dst] + 1	“N” set if result < 0 “Z” set if result = 0 “V” set if [Dst] was 077777
DEC	11110001	Dst ← [Dst] - 1	“N” set if result < 0 “Z” set if result = 0 “V” set if [Dst] was 100000
CLR	11110010	Dst ← 0	“N” reset “V” reset “C” reset “Z” set
INV	11110011	Dst ← INV([Dst])	“N” set if MSB of result = 1 “V” reset “C” set “Z” set if result = 0
LSR	11110100	Dst ← 0 || $[Dst]_{15->1}$	“N” set if MSB of result = 1 “V” = N XOR C “C” is loaded by last bit shifted out of register “Z” set if result = 0
ROR	11110101	Dst ← $[Dst]_0$ || $[Dst]_{15->1}$	“N” set if MSB of result = 1 “V” = N XOR C

			“C” is loaded by last bit shifted out of register “Z” set if result = 0
RRC	11110110	Dst ← C || $[Dst]_{15->1}$	“N” set if MSB of result = 1 “V” = N XOR C “C” is loaded by last bit shifted out of register “Z” set if result = 0
ASR	11110111	Dst ← $[Dst]_{15}$ || $[Dst]_{15->1}$	“N” set if MSB of result = 1 “V” = N XOR C “C” is loaded by last bit shifted out of register “Z” set if result = 0
LSL	11111000	Dst ← $[Dst]_{14->0}$ || 0	“N” set if MSB of result = 1 “V” = N XOR C “C” is loaded by last bit shifted out of register “Z” set if result = 0
ROL	11111001	Dst ← $[Dst]_{14->0}$ || $[Dst]_{15}$	“N” set if MSB of result = 1 “V” = N XOR C “C” is loaded by last bit shifted out of register “Z” set if result = 0
RL	11111010	Dst ← $[Dst]_{14->0}$ || C	“N” set if MSB of result = 1 “V” = N XOR C “C” is loaded by last bit shifted out of register “Z” set if result = 0

Branch instructions:

- ❏ Syntax “Opcode Offset”
- ❏ Operation “PC ← PC + Offset” is performed if branch condition is TRUE
- ❏ IR:

6bits opcode	10bits offset

Instruction Name	Op Code (Binary)	Operation Condition	Condition codes
BR	000000	No Condition	____
BEQ	000001	Z = 1	____
BNE	000010	Z = 0	____
BLO	000011	C = 0	____
BLS	110000	C = 0 OR Z = 1	____
BHI	110001	C = 1	____
BHS	110011	C = 1 OR Z = 1	____

No operation instructions:

- ❏ Syntax “Opcode”
- ❏ IR:

4bits opcode	___

Instruction Name	Op Code (Binary)	Operation Performed	Condition codes
HLT	1010	Stop the processor	____
NOP	1011	No operation is performed Continue code	____

Jump Subroutine instructions:

- ❏ Syntax “Opcode”
- ❏ IR:

4bits opcode	12bits ADDRESS

Instruction Name	Op Code (Binary)	Syntax	Condition codes
JSR	1101	JSR Address	___
RTS	111001	RTS	___

INTERRUPT	111010	___	___
IRET	111011	IRET	___

ALU Operations:

- ADD : F = Temp0 + Bus_{out}.
- SUB: F = Bus_{out} - Temp0.
- AND: F = Temp0 AND Bus_{out}.
- OR: F = Temp0 OR Bus_{out}.
- NOT: F = NOT Bus_{out}.
- XNOR: F = Temp0 XNOR Bus_{out}.
- LSR: F = LSR Bus_{out}.
- ROR: F = ROR Bus_{out}.
- RRC: F = RRC Bus_{out}.
- ASR: F = ASR Bus_{out}.
- LSL: F = LSL Bus_{out}.
- ROL: F = ROL Bus_{out}.
- RLC: F = ROL Bus_{out}.
- DEC: F = Bus_{out} - 1.
- INC: F = Bus_{out} + 1.
- OUTZero: F = 0.

Control Store Groups:

- Group 0 : “6-Bits”
  - Next Address Field.

- Group 1(out group): “3-Bits”
  - No Transfer.
  - $PC_{out.}$
  - $IR_{out.}$
  - $MDR_{out.}$
  - $Rsrc_{out.}$
  - $Rdest_{out.}$
  - $Temp1_{out.}$
  - $ALU_{out.}$

- Group 2 (in group): “3-Bits ”
  - No Transfer.
  - $PC_{in.}$
  - $IR_{in.}$
  - $Rsrc_{in.}$
  - $Rdest_{in.}$
  - $FLAGS_{in.}$

- Group 3(in group): “2-Bits”
  - No Transfer.
  - $MAR_{in.}$
  - $MDR_{in.}$

- ➢ $Temp1_{in.}$

❑ Group 4: “2-Bits”

- ➢ No Action.
- ➢ $Temp0_{in.}$
- ➢ Clear Temp0.

❑ Group 5: “3-Bits”

- ➢ No operation.
- ➢ $ALU_{add.}$
- ➢ $ALU_{bus+1.}$
- ➢ $ALU_{bus-1.}$
- ➢ $ALU_{op.}$

❑ Group 6: “2-Bits”

- ➢ Read.
- ➢ Write.
- ➢ No Action.

❑ Group 7: “1-Bit”

- ➢ Carry in = 1.
- ➢ Carry in = 0.

❑ Group 8: “1-Bit”

- ➢ No Action
- ➢ $PLA_{out.}$

- Group 9 (PLA Selectors): “3-Bits”
  - No Action.
  - ORdest
  - ORdest$_{indirect.}$
  - ORsrc$_{indirect.}$
  - ORresult.
  - ORop.
  - Address_field$_{out.}$
  - Flags$_{out.}$

Flag Register:

- Bit 0: Carry Flag.
- Bit 1: Zero Flag.
- Bit 2: Negative Flag.
- Bit 3: Parity Flag.
- Bit 4: Overflow Flag.

Addressing Modes:

- Register Direct: 000
- Auto Increment: 001
- Auto Decrement: 010
- Indexed: 011
- Register Indirect: 100
- Auto Increment Indirect: 101
- Auto Decrement Indirect: 110
- Indexed Indirect: 111

Control Store:

Address	Control Word	Next Address Field
000000	PC_{out} , MAR_{in} , Read , ALU_{bus+1}	000001
000001	ALU_{out}, PC_{in}	000010
000010	MDR_{out}, IR_{in}, uAR ← [PLA]	000011
000011	R_{srcout}, $Temp1_{in}$, uAR ← $[OR_{dest}]$	010000
000100	R_{srcout}, MAR_{in}, Read, uAR ← $[OR_{src.ind}]$	001110
000101	R_{srcout}, MAR_{in} , Read , ALU_{bus+1}	000110
000110	ALU_{out}, $R_{src.in}$, uAR ← $[OR_{src.ind}]$	001110
000111	R_{srcout}, ALU_{bus-1}	001000
001000	ALU_{out}, $R_{src.in}$, MAR_{in}, Read, uAR ← $[OR_{src.ind}]$	001110
001001	PC_{out} , MAR_{in} , Read , ALU_{bus+1}	001010
001010	ALU_{out}, PC_{in}	001011
001011	MDR_{out}, $Temp0_{in}$	001100
001100	R_{srcout}, ALU_{add}	001101
001101	ALU_{out}, MAR_{in}, Read, uAR ← $[OR_{src.ind}]$	001110
001110	MDR_{out}, MAR_{in}, Read	001111

001111	MDR_{out}, $Temp1_{in}$, uAR ← [OR_{dest}]	010000
010000	$R_{dest.out}$, $Temp0_{in}$, uAR ← [OR_{op}]	101110
010001	$R_{dest.out}$, MAR_{in}, Read, uAR ← [$OR_{dest.ind}$]	011011
010010	$R_{dest.out}$, MAR_{in} , Read , ALU_{bus+1}	010011
010011	ALU_{out}, $R_{dest.in}$, uAR ← [$OR_{dest.ind}$]	011011
010100	$R_{dest.out}$, ALU_{bus-1}	010101
010101	ALU_{out}, $R_{dest.in}$, MAR_{in}, Read, uAR ← [$OR_{dest.ind}$]	011011
010110	PC_{out} , MAR_{in} , Read , ALU_{bus+1}	010111
010111	ALU_{out}, PC_{in}	011000
011000	MDR_{out}, $Temp0_{in}$	011001
011001	$R_{dest.out}$, ALU_{add}	011010
011010	ALU_{out}, MAR_{in}, Read, uAR ← [$OR_{dest.ind}$]	011011
011011	MDR_{out}, MAR_{in}, Read	011100
011100	MDR_{out}, $Temp0_{in}$, uAR ← [OR_{op}]	101110
011101	R_{srcout}, ALU_{bus-1}	011110
011110	ALU_{out}, MAR_{in}, $R_{src.in}$	011111
011111	$Flags_{out}$, MDR_{in}, WR	100000
100000	R_{srcout}, ALU_{bus-1}	100001
100001	ALU_{out}, MAR_{in}, $R_{src.in}$	100010

100010	PC_{out} , MDR_{in} , WR, uAR ← [OR_{op}]	101110
100011	$Address\_field_{out}$, PC_{in}	000000
100100	R_{srcout}, MAR_{in}, Read, ALU_{bus+1}	100101
100101	ALU_{out}, $R_{src.in}$	100110
100110	MDR_{out} , PC_{in}, uAR ← [OR_{op}]	101110
100111	R_{srcout}, MAR_{in}, Read, ALU_{bus+1}	101000
101000	ALU_{out}, $R_{src.in}$	101001
101001	MDR_{out} , $FLAGS_{in}$	000000
101010	uAR ← [OR_{result}]	101011
101011	PC_{out}, $Temp0_{in}$	101100
101100	$Address\_field_{out}$, ALU_{add}	101101
101101	ALU_{out}, PC_{in}	000000
101110	$Temp1_{out}$, MDR_{in}, WR	000000
101111	$Temp1_{out}$, $R_{dest.in}$	000000
110000	$Temp1_{out}$, ALU_{op}, uAR ← [$OR_{dest.ind}$]	110001
110001	ALU_{out}, $R_{dest.in}$	000000
110010	ALU_{out}, MDR_{in}, WR	000000

CPU Cycles Analysis:

- Total CPU cycles = 9408
- Total MEM access = 2255
- Average MEM access = 3.331
- CPI = 13.897

	Clock Cycles		
Instruction	**Min**	**Max**	**Avg**
MOV	6	21	11
ADD ADC	7	23	14
SUB SBC	10	26	17
AND OR XNOR	7	23	14
CMP	9	25	16
INC DEC INV	7	15	10
CLR	6	12	8
LSR ROR RRC ASR	7	14	9

LSL ROL RLC			
BR	8		
BEQ BHI BHS BLO BLS BNE	9		
HLT INT IRET JSR X(R) NOP RTS	4		

Memory Access Analysis

Group 1: Two Operands Regular Instructions (ADD, ADC, SUB, SUBC, AND, OR, XNOR):

Addressing modes are (RGS: register, IND_RGS: indirect register)

SRC	DST	MA
RGS	RGS	1
RGS	IND_RGS	3
RGS	Autoincremet	3
RGS	IND_autoincremetn	4
RGS	Autodecrement	3
RGS	IND_Autodec	4
RGS	Indexed	4
RGS	Ind_Indexed	5
INC	RGS	2
INC	IND_RGS	4
INC	Autoincremet	4
INC	IND_autoincremetn	5
INC	Autodecrement	4
INC	IND_Autodec	5
INC	Indexed	5
INC	Ind_Indexed	6
DEC	RGS	2
DEC	IND_RGS	4
DEC	Autoincremet	4
DEC	IND_autoincremetn	5
DEC	Autodecrement	4
DEC	IND_Autodec	5
DEC	Indexed	5
DEC	Ind_Indexed	6
INDEX	RGS	3
INDEX	IND_RGS	5
INDEX	Autoincremet	5
INDEX	IND_autoincremetn	6
INDEX	Autodecrement	5
INDEX	IND_Autodec	6
INDEX	Indexed	6
INDEX	Ind_Indexed	7

SRC	DST	MA
IND_RGS	RGS	2
IND_RGS	IND_RGS	4
IND_RGS	Autoincremet	4
IND_RGS	IND_autoincremetn	5
IND_RGS	Autodecrement	4
IND_RGS	IND_Autodec	5
IND_RGS	Indexed	5
IND_RGS	Ind_Indexed	6
IND_INC	RGS	3
IND_INC	IND_RGS	5
IND_INC	Autoincremet	5
IND_INC	IND_autoincremetn	6
IND_INC	Autodecrement	5
IND_INC	IND_Autodec	6
IND_INC	Indexed	6
IND_INC	Ind_Indexed	7
IND_DEC	RGS	3
IND_DEC	IND_RGS	5
IND_DEC	Autoincremet	5
IND_DEC	IND_autoincremetn	6
IND_DEC	Autodecrement	5
IND_DEC	IND_Autodec	6
IND_DEC	Indexed	6
IND_DEC	Ind_Indexed	7
IND_INDEX	RGS	4
IND_INDEX	IND_RGS	6
IND_ INDEX	Autoincremet	6
IND_ INDEX	IND_autoincremetn	7
IND_ INDEX	Autodecrement	6
IND_ INDEX	IND_Autodec	7
IND_ INDEX	Indexed	7
IND_ INDEX	Ind_Indexed	8

Group 2: Two Operands Special Instructions (MOV, CMP):

As MOV doesn't require the final data of the destination, just the address, and CMP doesn't store the final value of the operation.

So, the second table would differ from the previous one, as MA would be less by 1, except when RGS is at the destination.

SRC	DST	MA
RGS	RGS	**1**
RGS	IND_RGS	2
RGS	Autoincremet	2
RGS	IND_autoincremetn	3
RGS	Autodecrement	2
RGS	IND_Autodec	3
RGS	Indexed	3
RGS	Ind_Indexed	4
INC	RGS	**2**
INC	IND_RGS	3
INC	Autoincremet	3
INC	IND_autoincremetn	4
INC	Autodecrement	3
INC	IND_Autodec	4
INC	Indexed	4
INC	Ind_Indexed	5
DEC	RGS	**2**
DEC	IND_RGS	3
DEC	Autoincremet	3
DEC	IND_autoincremetn	4
DEC	Autodecrement	3
DEC	IND_Autodec	4
DEC	Indexed	4
DEC	Ind_Indexed	5
INDEX	RGS	**3**
INDEX	IND_RGS	4
INDEX	Autoincremet	4
INDEX	IND_autoincremetn	5
INDEX	Autodecrement	4
INDEX	IND_Autodec	5
INDEX	Indexed	5
INDEX	Ind_Indexed	6

SRC	DST	MA
IND_RGS	RGS	**2**
IND_RGS	IND_RGS	3
IND_RGS	Autoincremet	3
IND_RGS	IND_autoincremetn	4
IND_RGS	Autodecrement	3
IND_RGS	IND_Autodec	4
IND_RGS	Indexed	4
IND_RGS	Ind_Indexed	5
IND_INC	RGS	**3**
IND_INC	IND_RGS	4
IND_INC	Autoincremet	4
IND_INC	IND_autoincremetn	5
IND_INC	Autodecrement	4
IND_INC	IND_Autodec	5
IND_INC	Indexed	5
IND_INC	Ind_Indexed	6
IND_DEC	RGS	**3**
IND_DEC	IND_RGS	4
IND_DEC	Autoincremet	4
IND_DEC	IND_autoincremetn	5
IND_DEC	Autodecrement	4
IND_DEC	IND_Autodec	5
IND_DEC	Indexed	5
IND_DEC	Ind_Indexed	6
IND_INDEX	RGS	**4**
IND_INDEX	IND_RGS	5
IND_ INDEX	Autoincremet	5
IND_ INDEX	IND_autoincremetn	6
IND_ INDEX	Autodecrement	5
IND_ INDEX	IND_Autodec	6
IND_ INDEX	Indexed	6
IND_ INDEX	Ind_Indexed	7

Group3: One Operand Instructions (INC, DEC, CLR, INV, LSR, ROR, RRC, ASR, LSL, ROL, RLC):

Destination	**MA**
Register	**1**
Indirect Register	3
Autoincremet	3
Indirect Autoincremetn	4
Autodecrement	3
Indirect Autodecrement	4
Indexed	4
Indirect Indexed	5

Group 4: Jumpers and Stackers (JSR, RTS, ITR, IRET):

Operation	Memory Access
JSR	3
RTS	2
ITR	3
IRET	3

Group 5: Branches, HLT and NO Operation:

Memory Access = 1 for fetching any of these instructions.