Cairo University

Faculty of Engineering

Architecture Project

Team 9

Team members:

Ahmed Salama Hamed

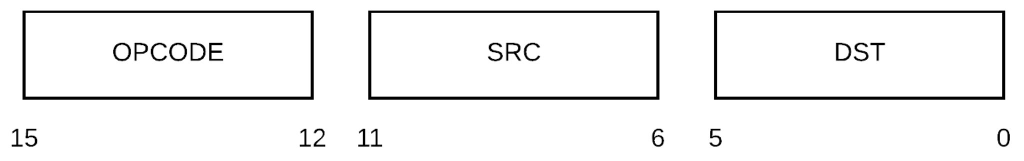
Ramy Said

Shehab Alaa

Mohamed Talaat

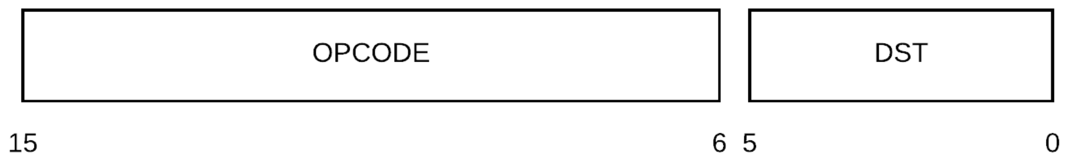
● Instruction Set

1. 2 Operand Instructions



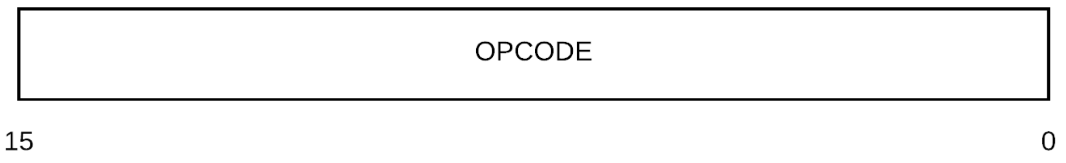
inst_num	Instruction	OPCODE(OCT)	OPeration
0	MOV	00	Dst ← [Src]
1	ADD	01	Dst ← [Dst] + [Src]
2	ADC	02	Dst ← [Dst] + [Src] + C
3	SUB	03	Dst ← [Dst] – [Src]
4	SBC	04	Dst ← [Dst] – [Src] – C
5	AND	05	Dst ← [Dst] AND [Src]
6	OR	06	Dst ← [Dst] OR [Src]
7	XNOR	07	Dst ← [Dst] XNOR [Src]
8	CMP	10	[Dst] – [Src]

2. 1 Operand Instructions



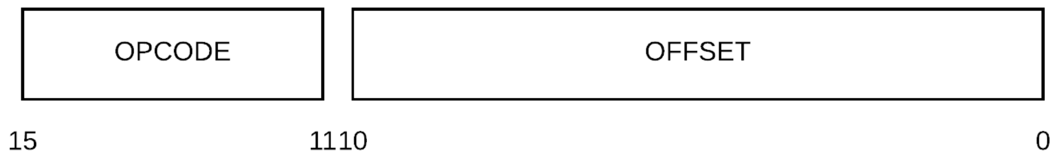
inst_num	Instruction	OPCODE(OCT)	OPeration
9	INC	1100	Dst ⟵ [Dst] + 1
10	DEC	1101	Dst ⟵ [Dst] – 1
11	CLR	1102	Dst ⟵ 0
12	INV	1103	Dst ⟵ INV([Dst])
13	LSR	1104	Dst ⟵ 0 || [Dst]15->1
14	ROR	1105	Dst ⟵ [Dst]0 || [Dst]15->1
15	RRC	1106	Dst ⟵ C || [Dst]15->1
16	ASR	1107	Dst ⟵ [Dst]15 || [Dst]15->1
17	LSL	1110	Dst ⟵ [Dst]14->0 || 0
18	ROL	1111	Dst ⟵ [Dst]14->0 || [Dst]15
19	RLC	1112	Dst ⟵ [Dst]14->0 || C

3. No Operand Instructions



inst_num	Instruction	OPCODE(OCT)
20	HLT	120000
21	NOP	121000

4. Branch Instructions



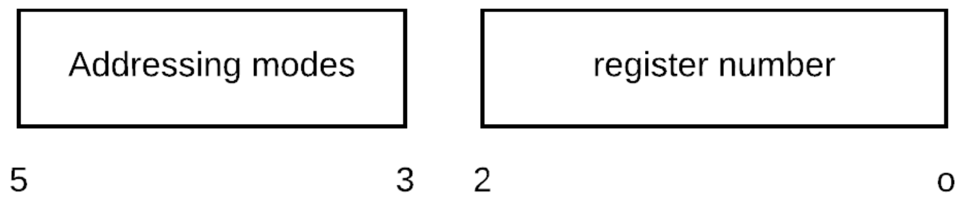
inst_num	Instruction	OPCODE(OCT)	Condition
22	BR	30	None
23	BEQ	31	Z=1
24	BNE	32	Z=0
25	BLO	33	C=0
26	BLS	34	C=0 or Z=1
27	BHI	35	C=1
28	BHS	36	C=1 or Z=1

5. Jump To Subroutine Instructions

6.

inst_num	Instruction	Syntax	IR	OPCODE(OCT)
29	JSR (Jump to subroutine)	jsr address	opcode(15-6) + dst(5-0)	1300
30	RTS	RTS	opcode(15-0)	130100
31	IRET		opcode(15-0)	130300
	INTERRUPT (Hardware Interrupt)	______________ ____	______________ ____	______________ ____

✗ Operand

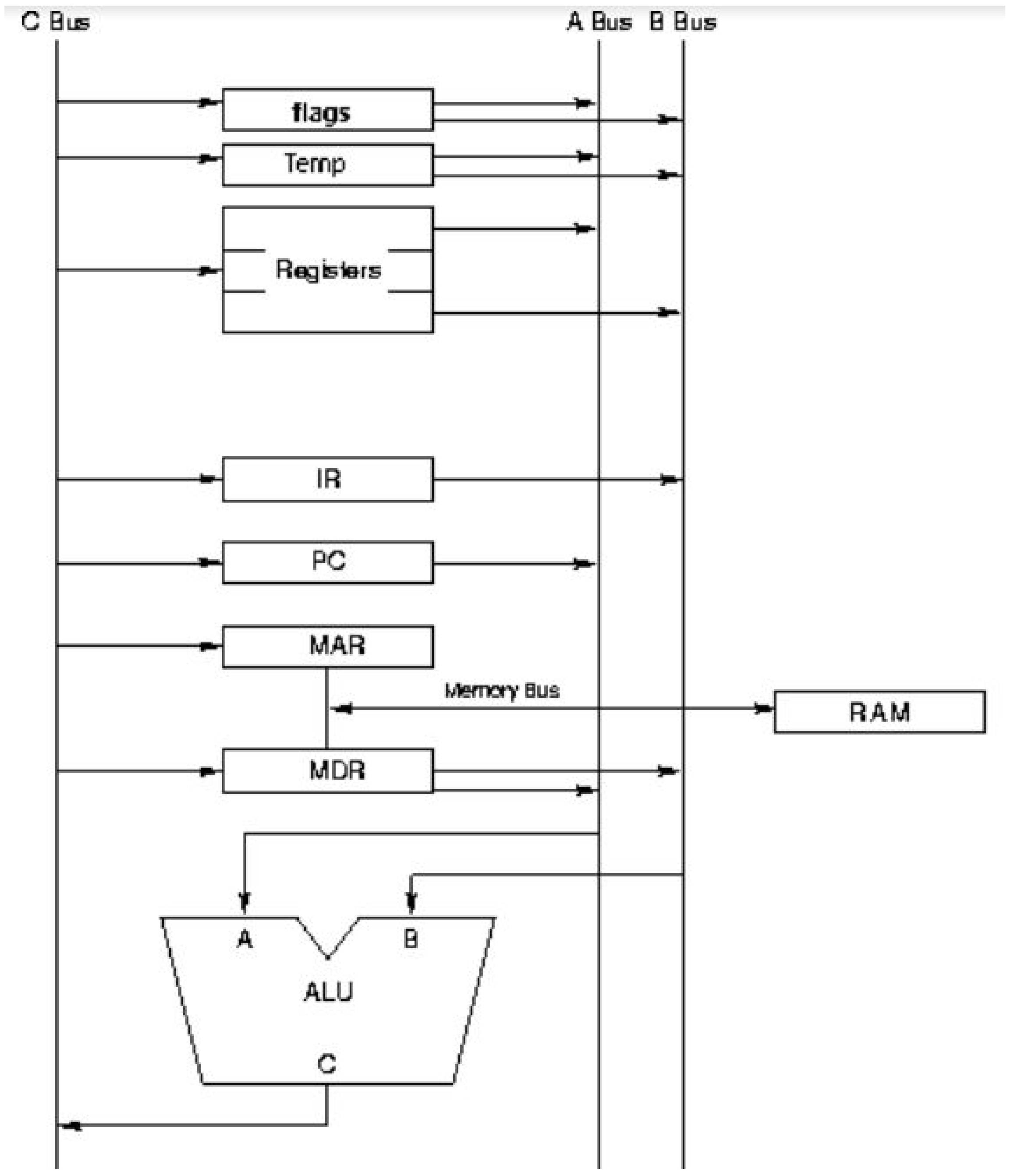


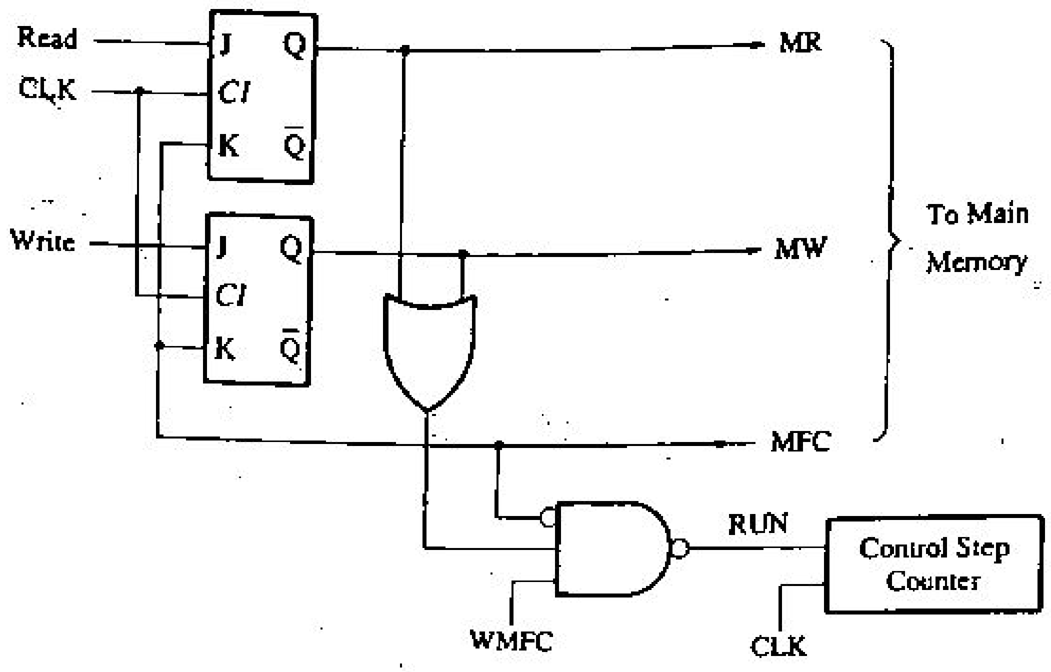
Register	Code(OCT)
R0	0
R1	1
R2	2
R3	3
R4	4

R5	5
R6	6
R7	7

Addressing Mode	Code(OCT)
Rn	0
(Rn)+	1
-(Rn)	2
x(Rn)	3
@Rn	4
@(Rn)+	5
@-(Rn)	6
@x(Rn)	7

- **Bus System**


C Bus
A Bus
B Bus
flags
Temp
Registers
IR
PC
MAR
Memory Bus
RAM
MDR
A
B
ALU
C


Read
CLK
Write
J
Cl
K
Q
$\overline{Q}$
J
Cl
K
Q
$\overline{Q}$
MR
MW
MFC
To Main Memory
RUN
Control Step Counter
WMFC
CLK

• Micro Instructions (examples)

General Micro Instructions	Addressing mode	Micro Instructions	MAs	Cycles
fetch /decode	Same for all	0.PCoutA,F=A,MARin,RD 1.PCoutA,INCA,PCin,WMFC 2.MDRoutA,F=A,IRin	1	3
fetch operand	Register mode "R0"	Rout ,Tempin	0	1
	Auto-increment "(R0)+"	0.R0outA,F=A,MARin,RD 1.R0outA,INC,R0in,WMFC 2.MDRout,F=A,TMPin	1	3
	Auto-decrement "-(R0)"	0.R0outA,dec,R0in,MARin,RD ,WMFC 1.MDRout,F=A,TMPin	1	2
	Indexed "X(R0)"	3.PCoutA,F=A,MARin,RD 4.PCoutA,A+1,PCin,WMFC 5.MDRoutA,R0outB,A+B, MARin,RD,WMFC 6.MDRoutA,F=A,Tempin //6->if 1st operand	2	4
	Register mode indirect "@R0"	3.R0outA,F=A,MARin,RD, WMFC 4.MDRoutA,F=A,Tempin //4->if 1st operand	1	2

	Auto-increment indirect “@(R0)+	3-RoutA , transferA,MARin ,Read 4-Routb ,B+1 ,Rin,WMFC 5-MDRoutA,transferA,MARin ,read ,WMFC, 6-mdrout ,tempin	2	4
	Auto-decrement indirect “@-(R0)”	3-Routb ,B-1 ,Rin, MARin ,Read ,WMFC 4-MDRoutA,transferA,MARin,read,wmfc 5-mdrout ,tempin	2	3
	Indexed indirect “@X(R0)”	3.PCoutA,F=A,MARin,RD 4.PCoutA,A+1,PCin,WMFC 5.MDRotA,R0outB,A+B,MARin, RD,WMFC 6.MDRoutA,F=A,MARin,RD, WMFC 7..MDRoutA,F=A,Tempin //7->if 1st operand	3	5

operation	add,adc,sub,subc, or,and,xnor,cmp	op1 outA, op2 outB
		B=A+B
		B=A+B+C
		B=B-A
		B=B-A-C
		B=B || A
		B=B.A
		B=A XNOR B
		B-A		

	mov	op1 out ,op2 in		

	inc,dec,inv,clr,lsr, ror,rrc,asr,lsl,rol ,rlc	op1 outA
		A = A+1
		A = A-1
		A = 0
		A= not A
		A =rotate A
		
		A=rotate A with Carry		

	br	addr of ir outB, PCoutA, F=A+B,pcin	0	~~1~~

	beq,bne,blo,bls,bhi,bhs	if condition
		addr of ir outB, PCoutA, F=A+B,pcin
		else end
				~~2~~

	hlt,nop			
save	Register mode “R0”	op1 ,op2 ,operation Rin	0	1
	Auto-increment “(R0)+”	0. --op1 outA,F=A,MDRin,WRITE ,WMFC,END	1	1
	Auto-decrement “-(R0)”	0. --op1 outA,F=A,MDRin,WRITE ,WMFC,END	1	1
	Indexed “X(R0)”	--op1 outA, op2 outB, perform operation ,MDRin,write,WMFC,end	1	1
	Register mode indirect “@R0”	--op1 outA, op2 outB, perform operation ,MDRin,write,WMFC,end	1	1
	Auto-increment indirect “@(R0)+	op1 ,op2 ,operation ,MDRin,write,WMFC,end	1	1
	Auto-decrement indirect “@-(R0)”	op1 ,op2 ,operation ,MDRin,write,WMFC,end	1	1
	Indexed indirect “@X(R0)”	--op1 outA, op2outB, perform operation ,MDRin,write,WMFC,end	1	1

Category	instruction (s)	addressing mode op1	addressing mode op2	Micro Instructions	MAs	Cycles
Two Operands Intruction EX:	add @x(R0), @y(R1)	Indexed indirect	Indexed indirect	fetch inst fetch op1 fetch op2 alu operation save from previous table: 0.PCoutA,F=A,MARin,RD 1.PCoutA,INCA,PCin,WMFC 2.MDRoutA,F=A,IRin 3. 0 4. 1 5.MDRotA,R0outB,A+B,MARin,RD,WMFC 6.MDRoutA,F=A,MARin,RD,WMFC 7.MDRoutA,F=A,Tempin 8. 0 9. 1 10.MDRotA,R1outB,A+B,MARin,RD,WMFC 11. 6 12.MDRoutA,TempoutB,A+B,MDRin 13.WRITE,WMFC,END	8	14
One Operand Intruction EX:	INC @X(R0)	Indexed indirect	_______________	fetch inst fetch op alu operation save from previous table: 0.PCoutA,F=A,MARin,RD 1.PCoutA,INCA,PCin,WMFC 2.MDRoutA,F=A,IRin 3. 0 4. 1 5.MDRotA,R0outB,A+B,MARin,RD,WMFC 6.MDRoutA,F=A,MARin,RD,WMFC	5	9

				7.MDRoutA,INCA,MDRin 8.WRITE,WMFC,END		
Branch Instruction EX:	BHIlable1	________________	________________	fetch inst fetch offset add offset ,pc and check save in pc 0.PCoutA,F=A,MARin,RD 1.PCoutA,INCA,PCin,WMFC 2.MDRoutA,F=A,IRin 3.PCoutA,IRoffsetoutB, F=A+B,TMPin, if {C!='1'} Then End 4.TMPoutA,PCin,END	1	5
No Operand Instruction EX:	HLT	________________	________________		1	3
JSR	JSR subroutine = JSR X(PC)	________________	________________	fetch instruction fetch address save in stack pc pc =address 0.PCoutA,F=A,MARin,RD 1.PCoutA,INCA,PCin,WMFC 2.MDRoutA,F=A,IRin 3..PCoutA,F=A,MARin,RD 4.PCoutA,INCA,PCin,WMFC 5.MDRouta,PCout B,F=A+B, tempin 6- SPout a ,F=A-1 , MARin,SPin, 7- Pcout a,F=A,MDRin,WR 8-temoutA,F=A,PC in, wmfc,end	3	9

RTS	RTS	________________	________________	0.PCoutA,F=A,MARin,RD 1.PCoutA,INCA,PCin,WMFC 2.MDRoutA,F=A,IRin 3- SPout a ,F=A , MARin,rd 4-SPout a , F=A+1, SPin ,wmfc 5-MDRout A , F=A , PCin,End	2	6
INTERRUPT	__________	________________	________________	0- SPout a ,F=A-1 , MARin,SPin, 1- Pcout a,F=A,MDRin,WR,WMFC 2- SPout a ,F=A-1 , MARin,SPin, 3- FlagoutA,F=A,MDRin,WR, WMFC 4-INT Addressout A, F=A,PC in ,end	2	5
IRET				0.PCoutA,F=A,MARin,RD 1.PCoutA,INCA,PCin,WMFC 2.MDRoutA,F=A,IRin 3- SPout a ,F=A , MARin,rd 4-SPout a , F=A+1, SPin ,wmfc 5-MDRoutA,F=A,FlagIn, 6- SPout a ,F=A , MARin,rd 7-SPout a , F=A+1, SPin,wmfc 8-MDRoutA,F=A,PCin,end	3	9

● Analysis

op1(addressing mode)	op2(addressing mode)	instructions(two operands)	MAs	Cycles
Register mode "R0"	Register mode "R1"	add,adc,sub,subc,or,and,xnor,cmp	1	4
		mov	1	3
Auto-increment "(R0)+"	Register mode "R1"	add,adc,sub,subc,or,and,xnor,cmp	3	6
		mov	2	6
Auto-decrement "-(R0)"	Register mode "R1"	add,adc,sub,subc,or,and,xnor,cmp	3	5
		mov	2	5
Indexed "X(R0)"	Register mode "R1"	add,adc,sub,subc,or,and,xnor,cmp	4	7
		mov	3	7
Register mode indirect "@R0"	Register mode "R1"	add,adc,sub,subc,or,and,xnor,cmp	3	5
		mov	2	5
Auto-increment indirect "@(R0)+	Register mode "R1"	add,adc,sub,subc,or,and,xnor,cmp	4	6
		mov	3	7
Auto-decrement indirect "@-(R0)"	Register mode "R1"	add,adc,sub,subc,or,and,xnor,cmp	4	6

		mov	3	7
Indexed indirect “@X(R0)”	Register mode “R1”	add,adc,sub,subc,or,and,xnor,cmp	5	8
		mov	4	8
Register mode “R0”	Auto-increment “(R1)+”	add,adc,sub,subc,or,and,xnor,cmp	2	6
		mov	2	6
Auto-increment “(R0)+”	Auto-increment “(R1)+”	add,adc,sub,subc,or,and,xnor,cmp	4	9
		mov	3	9
Auto-decrement “-(R0)”	Auto-increment “(R1)+”	add,adc,sub,subc,or,and,xnor,cmp	4	8
		mov	3	8
Indexed “X(R0)”	Auto-increment “(R1)+”	add,adc,sub,subc,or,and,xnor,cmp	5	10
		mov	4	10
Register mode indirect “@R0”	Auto-increment “(R1)+”	add,adc,sub,subc,or,and,xnor,cmp	4	8
		mov	3	8
Auto-increment indirect “@(R0)+	Auto-increment “(R1)+”	add,adc,sub,subc,or,and,xnor,cmp	5	10
		mov	4	10
Auto-decrement indirect “@-(R0)”	Auto-increment “(R1)+”	add,adc,sub,subc,or,and,xnor,cmp	5	9
		mov	4	9

Indexed indirect “@X(R0)”	Auto-increment “(R1)+”	add,adc,sub,subc,or,and,xnor,cmp	6	11
		mov	5	10
Register mode “R0”	Auto-decrement “-(R1)”	add,adc,sub,subc,or,and,xnor,cmp	2	5
		mov	2	5
Auto-increment “(R0)+”	Auto-decrement “-(R1)”	add,adc,sub,subc,or,and,xnor,cmp	4	8
		mov	3	8
Auto-decrement “-(R0)”	Auto-decrement “-(R1)”	add,adc,sub,subc,or,and,xnor,cmp	4	7
		mov	3	7
Indexed “X(R0)”	Auto-decrement “-(R1)”	add,adc,sub,subc,or,and,xnor,cmp	5	9
		mov	4	9
Register mode indirect “@R0”	Auto-decrement “-(R1)”	add,adc,sub,subc,or,and,xnor,cmp	4	7
		mov	3	7
Auto-increment indirect “@(R0)+	Auto-decrement “-(R1)”	add,adc,sub,subc,or,and,xnor,cmp	5	9
		mov	4	9
Auto-decrement indirect “@-(R0)”	Auto-decrement “-(R1)”	add,adc,sub,subc,or,and,xnor,cmp	5	8
		mov	4	9
Indexed indirect “@X(R0)”	Auto-decrement “-(R1)”	add,adc,sub,subc,or,and,xnor,cmp	6	10

		mov	5	10
Register mode “R0”	Indexed “X(R1)”	add,adc,sub,subc,or,and,xnor,cmp	3	7
		mov	3	7
Auto-increment “(R0)+”	Indexed “X(R1)”	add,adc,sub,subc,or,and,xnor,cmp	5	10
		mov	4	10
Auto-decrement “-(R0)”	Indexed “X(R1)”	add,adc,sub,subc,or,and,xnor,cmp	5	9
		mov	4	9
Indexed “X(R0)”	Indexed “X(R1)”	add,adc,sub,subc,or,and,xnor,cmp	6	11
		mov	5	11
Register mode indirect “@R0”	Indexed “X(R1)”	add,adc,sub,subc,or,and,xnor,cmp	5	9
		mov	4	9
Auto-increment indirect “@(R0)+	Indexed “X(R1)”	add,adc,sub,subc,or,and,xnor,cmp	6	11
		mov	5	11

Auto-decrement indirect “@-(R0)”	Indexed “X(R1)”	add,adc,sub,subc,or,and,xnor,cmp	6	10
		mov	5	10
Indexed indirect “@X(R0)”	Indexed “X(R1)”	add,adc,sub,subc,or,and,xnor,cmp	7	12
		mov	6	12
Register mode “R0”	Register mode indirect “@R1”	add,adc,sub,subc,or,and,xnor,cmp	2	5
		mov	2	5
Auto-increment “(R0)+”	Register mode indirect “@R1”	add,adc,sub,subc,or,and,xnor,cmp	4	8
		mov	3	8
Auto-decrement “-(R0)”	Register mode indirect “@R1”	add,adc,sub,subc,or,and,xnor,cmp	4	7
		mov	3	7
Indexed “X(R0)”	Register mode indirect “@R1”	add,adc,sub,subc,or,and,xnor,cmp	5	9
		mov	4	9
Register mode indirect “@R0”	Register mode indirect “@R1”	add,adc,sub,subc,or,and,xnor,cmp	4	7
		mov	3	7
Auto-increment indirect “@(R0)+	Register mode indirect “@R1”	add,adc,sub,subc,or,and,xnor,cmp	5	9
		mov	4	9

Auto-decrement indirect “@-(R0)”	Register mode indirect “@R1”	add,adc,sub,subc,or,and,xnor,cmp	5	8
		mov	4	8
Indexed indirect “@X(R0)”	Register mode indirect “@R1”	add,adc,sub,subc,or,and,xnor,cmp	6	10
		mov	5	10
Register mode “R0”	Auto-increment indirect “@(R0)+	add,adc,sub,subc,or,and,xnor,cmp	3	7
		mov	2	7
Auto-increment “(R0)+”	Auto-increment indirect “@(R0)+	add,adc,sub,subc,or,and,xnor,cmp	5	10
		mov	4	10
Auto-decrement “-(R0)”	Auto-increment indirect “@(R0)+	add,adc,sub,subc,or,and,xnor,cmp	5	9
		mov	4	9
Indexed “X(R0)”	Auto-increment indirect “@(R0)+	add,adc,sub,subc,or,and,xnor,cmp	6	11
		mov	5	11
Register mode indirect “@R0”	Auto-increment indirect “@(R0)+	add,adc,sub,subc,or,and,xnor,cmp	5	9
		mov	4	9

Auto-increment indirect “@(R0)+	Auto-increment indirect “@(R0)+	add,adc,sub,subc,or,and,xnor,cmp	6	11
		mov	5	11
Auto-decrement indirect “@-(R0)”	Auto-increment indirect “@(R0)+	add,adc,sub,subc,or,and,xnor,cmp	6	10
		mov	5	10
Indexed indirect “@X(R0)”	Auto-increment indirect “@(R0)+	add,adc,sub,subc,or,and,xnor,cmp	7	12
		mov	6	12
Register mode “R0”	Auto-decrement indirect “@-(R0)”	add,adc,sub,subc,or,and,xnor,cmp	3	6
		mov	3	6
Auto-increment “(R0)+”	Auto-decrement indirect “@-(R0)”	add,adc,sub,subc,or,and,xnor,cmp	5	9
		mov	4	9
Auto-decrement “-(R0)”	Auto-decrement indirect “@-(R0)”	add,adc,sub,subc,or,and,xnor,cmp	5	8
		mov	4	8
Indexed “X(R0)”	Auto-decrement indirect “@-(R0)”	add,adc,sub,subc,or,and,xnor,cmp	6	10
		mov	5	10

Register mode indirect “@R0”	Auto-decrement indirect “@-(R0)”	add,adc,sub,subc,or,and,xnor,cmp	5	8
		mov	4	8
Auto-increment indirect “@(R0)+	Auto-decrement indirect “@-(R0)”	add,adc,sub,subc,or,and,xnor,cmp	6	10
		mov	5	10
Auto-decrement indirect “@-(R0)”	Auto-decrement indirect “@-(R0)”	add,adc,sub,subc,or,and,xnor,cmp	6	9
		mov	5	9
Indexed indirect “@X(R0)”	Auto-decrement indirect “@-(R0)”	add,adc,sub,subc,or,and,xnor,cmp	7	11
		mov	6	11
Register mode “R0”	Indexed indirect “@X(R0)”	add,adc,sub,subc,or,and,xnor,cmp	4	8
		mov	3	8
Auto-increment “(R0)+”	Indexed indirect “@X(R0)”	add,adc,sub,subc,or,and,xnor,cmp	6	11
		mov	5	11
Auto-decrement “-(R0)”	Indexed indirect “@X(R0)”	add,adc,sub,subc,or,and,xnor,cmp	6	10
		mov	5	10

Indexed "X(R0)"	Indexed indirect "@X(R0)"	add,adc,sub,subc,or,and,xnor,cmp	7	12
		mov	6	12
Register mode indirect "@R0"	Indexed indirect "@X(R0)"	add,adc,sub,subc,or,and,xnor,cmp	6	10
		mov	5	10
Auto-increment indirect "@(R0)+	Indexed indirect "@X(R0)"	add,adc,sub,subc,or,and,xnor,cmp	7	12
		mov	6	12
Auto-decrement indirect "@-(R0)"	Indexed indirect "@X(R0)"	add,adc,sub,subc,or,and,xnor,cmp	7	11
		mov	6	11
Indexed indirect "@X(R0)"	Indexed indirect "@X(R0)"	add,adc,sub,subc,or,and,xnor	8	13
		mov	7	13
SUM		add,adc,sub,subc,or,and,xnor ,cmp	7*(**27+35+35+43+35+43+43+51 = 312) +(19+27+27+35+27+35+35+43=248)**	8*(**48+71+63+79+63+79+71+87) =8*(561)=4488**
		mov	254	556

op(addressing mode)	instructions	MAs	Cycles
Register mode “R0”	INC,DEC,LSR,ROR,RRC,ASR,LSL,ROL,RLC	1	4
	CLR	1	4
Auto-increment “(R0)+”	INC,DEC,LSR,ROR,RRC,ASR,LSL,ROL,RLC	3	6
	CLR	2	5
Auto-decrement “-(R0)”	INC,DEC,LSR,ROR,RRC,ASR,LSL,ROL,RLC	3	6
	CLR	2	5
Indexed “X(R0)”	INC,DEC,LSR,ROR,RRC,ASR,LSL,ROL,RLC	4	7
	CLR	3	7
Register mode indirect “@R0”	INC,DEC,LSR,ROR,RRC,ASR,LSL,ROL,RLC	3	5
	CLR	2	5
Auto-increment indirect “@(R0)+	INC,DEC,LSR,ROR,RRC,ASR,LSL,ROL,RLC	4	7
	CLR	3	7
Auto-decrement indirect “@-(R0)”	INC,DEC,LSR,ROR,RRC,ASR,LSL,ROL,RLC	4	7
	CLR	3	7
Indexed indirect “@X(R0)”	INC,DEC,LSR,ROR,RRC,ASR,LSL,ROL,RLC	5	8

	CLR	4	8
SUM		9*(1+3+3 +4+3+4 +4+5) +(1+2+2+ 3+2+3 +3+4) = 263	9*(4+6+6 +7+5+7 +7+8) +(4+5+5 +7+5+7 +7+8) = 498

Instructions	MA’s	Cycles	Total Cycles
All Branch instruction	1	5 except (Br =4)	5*6+4=34

Instructions	MA’s	Cycles	Total Cycles
No Operand	1	3	6

Instructions	MA’s	Cycles
RTS	2	6
JSR	3	9
INTERRUPT	2	5
IRET	3	10

Total Cycles = 4488+556+498+34+6+6+9+5+10 = 5614

CPI = 5614/677 = 8.292

• Assembler Program

• Files

1. Source Script.
2. Codes.txt (must be in the same directory with source script) that have all binary codes of instructions.

• Assumptions

1. labels and variables are case sensitive.
2. opcodes are not case sensitive (INC = inc = InC .....).
3. assume JSR Label only.

• Arguments Format

1. input file.
2. output file.
3. debug file (to make debug process easy).