

Laboratory Report

2022

COMPUTER ORGANISATION AND ARCHITECTURE (EET 2211)



Name:

Registration No:

Branch:.....Section:.....

CONTENTS

S. No.	Name of the Experiment	Date of Experiment	Page No.	Remark
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				

CONTENTS

S. No.	Name of the Experiment	Date of Experiment	Page No.	Remark
1	Examine and analyze different addressing mode of 8086 microprocessor	18.05.2022		
2	Evaluate different arithmetic operations on two 16 bit data	25.05.2022		
3	Evaluate different logical operation on 16-bit data.	08.06.2022		
4	Analyze and Evaluate branch operations of 8086 microprocessor on 8/16-bit data.	22.06.2022		
5	Analyze and evaluate different array operation using 8086 microprocessor.	29.06.2022		
6	Evaluate Different Arithmetic Operation & Logical operations on two 32 bit data using ARM processor.	06.07.2022		
7	Analyze & evaluate different array operation on 32 bit data using ARM processor.	13.07.2022		
8				
9				
10				

Observations (with screen shots):

Objective 1:

The screenshot shows a debugger interface with the following components:

- Assembly Editor:** Displays the assembly code:

```
01 ORG 100H
02 MOU AX, 1234H
03 MOU BX, 5678H
04 ADD AX, BX
05 HLT
```
- Registers Window:** Shows the state of various registers:

Registers	H	L	Value
AX	68	AC	01234h
BX	56	78	05678h
CX	00	09	00009h
DX	00	00	00000h
CS	0700		F4 244h
IP	0108		98 144h
SS	0700		98 144h
SP	FFFE		98 144h
BP	0000		98 144h
SI	0000		98 144h
DI	0000		98 144h
DS	0700		98 144h
ES	0700		98 144h
- Memory Dump:** A window titled "Random Access Memory" showing memory starting at address 0700:0108. The data is mostly zeros with some high-value bytes (e.g., FF, 98) appearing.
- Emulator Window:** Shows the instruction flow from 07100 to 07115. The instruction at 07108 (MOU AX, 1234H) is highlighted.

Objective 2:

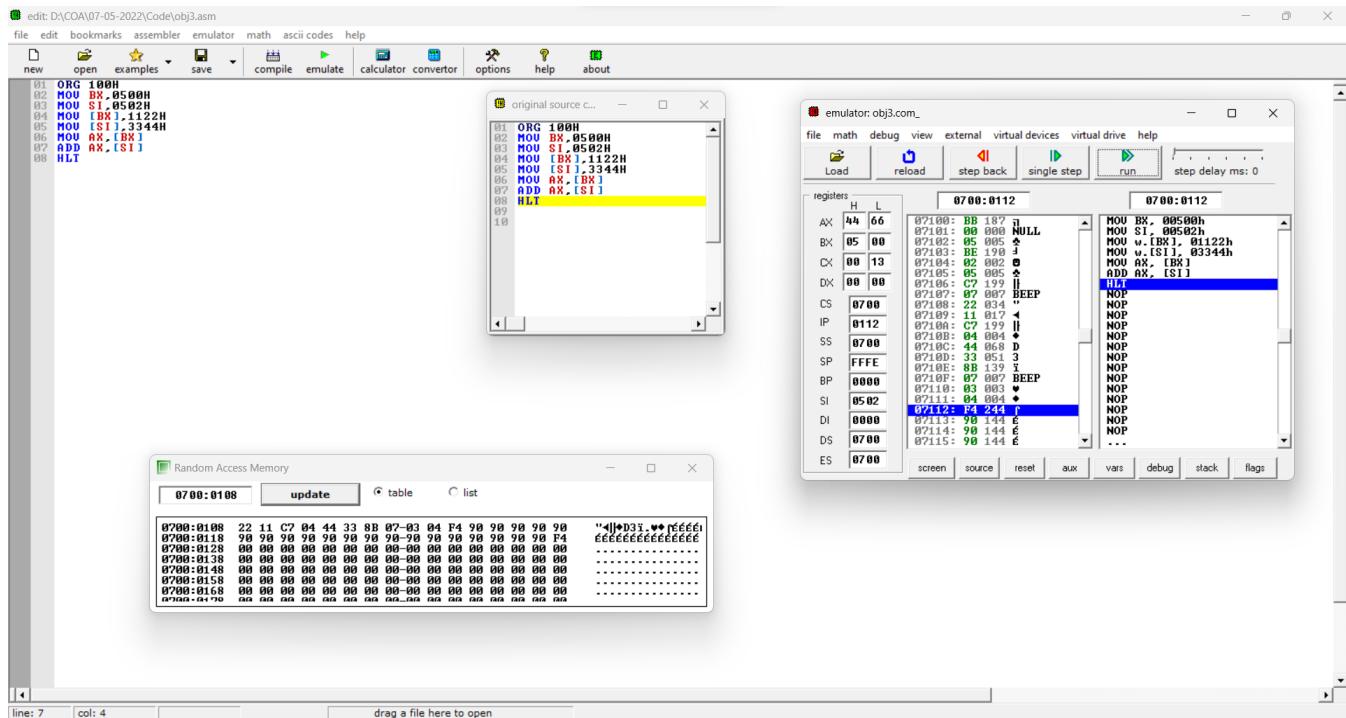
The screenshot shows a debugger interface with the following components:

- Assembly Editor:** Displays the assembly code:

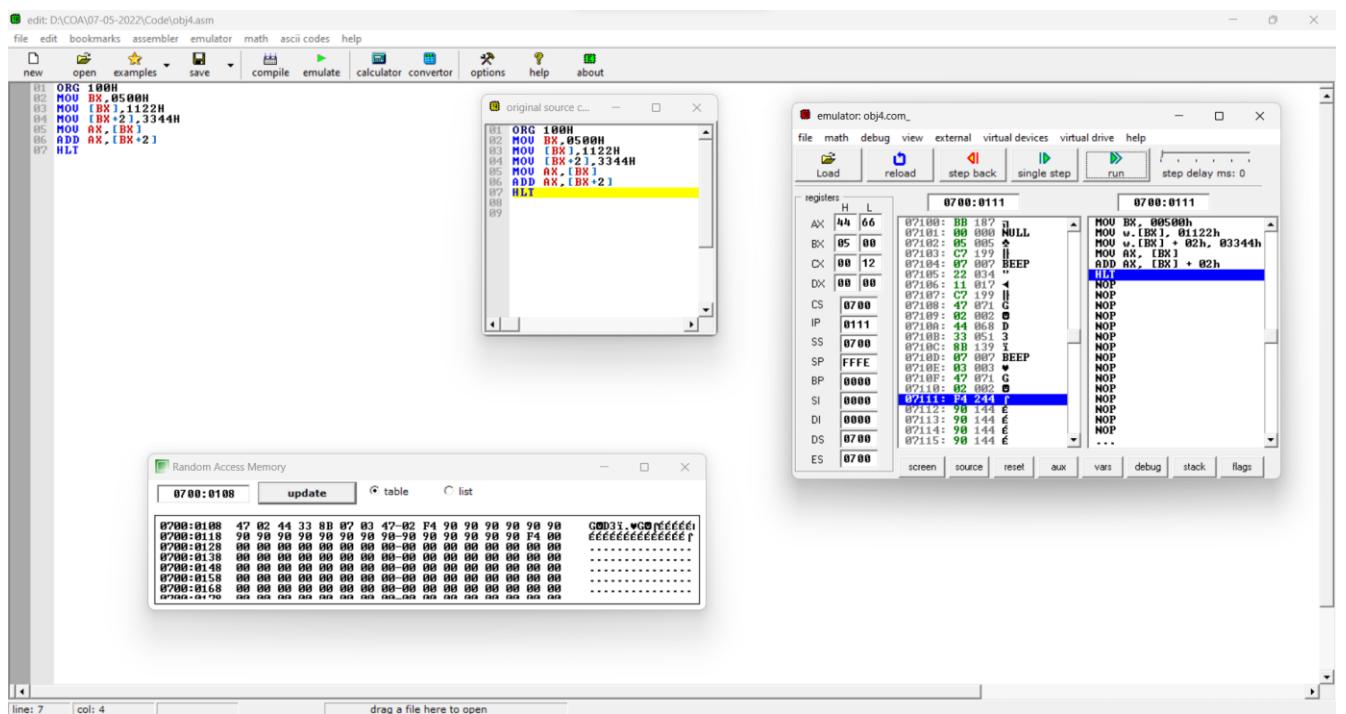
```
01 ORG 100H
02 MOU [500H], 1122H
03 MOU [502H], 0344H
04 ADD AX, [500H]
05 HLT
```
- Registers Window:** Shows the state of various registers:

Registers	H	L	Value
AX	44	66	C2 199h
BX	00	00	00 000h NULL
CX	00	14	00 005h
DX	00	00	22 017h
CS	0700		97 005h
IP	0113		44 068h
SS	0700		33 051h
SP	FFFE		00 000h NULL
BP	0000		03 003h
SI	0000		02 002h
DI	0000		01 001h
DS	0700		98 144h
ES	0700		98 144h
- Memory Dump:** A window titled "Random Access Memory" showing memory starting at address 0700:0108. The data is mostly zeros with some high-value bytes (e.g., FF, 98) appearing.
- Emulator Window:** Shows the instruction flow from 07100 to 07115. The instruction at 07113 (HLT) is highlighted.

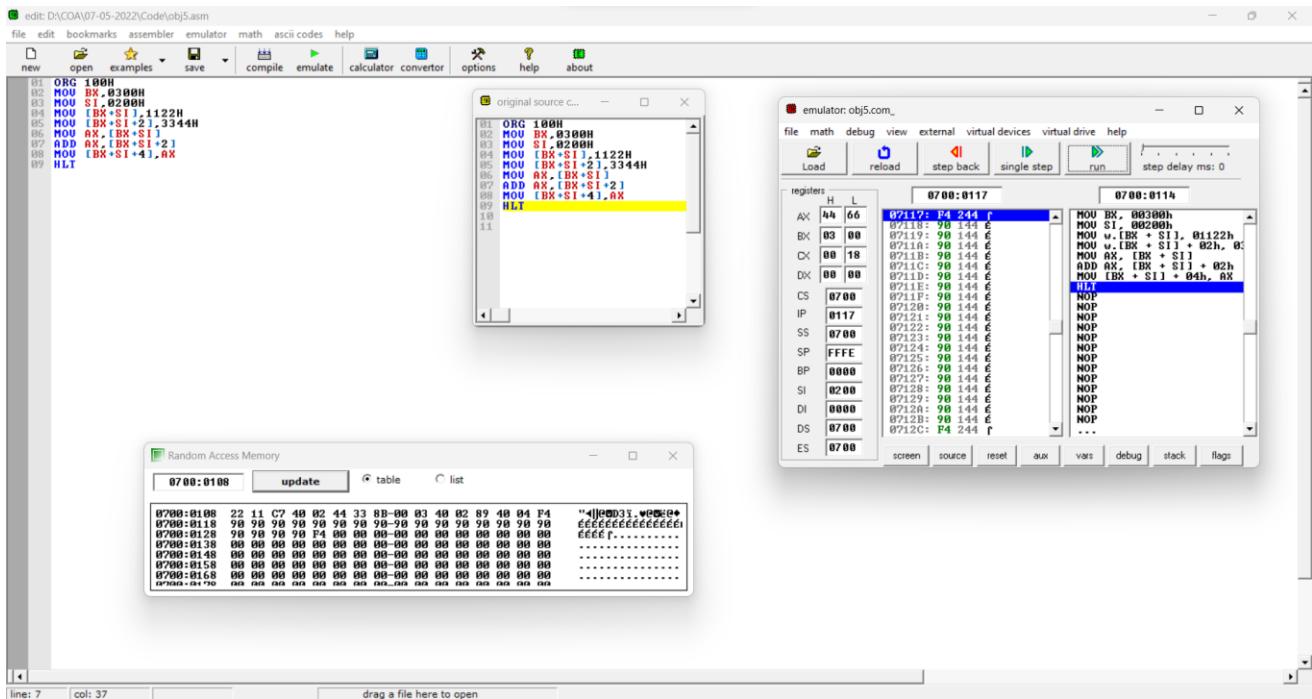
Objective 3:



Objective 4:



Objective 5:



Observations (with screen shots):

For Obj. 1

The screenshot shows a debugger interface with the following components:

- Assembly Editor:** Displays the assembly code for obj.1.asm. The code includes instructions like mov ax, 1000h, mov ds, ax, mov ax, [5000h], mov bx, [5002h], add ax, bx, mov [5004h], ax, and hlt.
- Registers Window:** Shows the state of CPU registers. AX has value 68, BX has 56, CX has 00 00, DX has 00 00, CS has 0100, IP has 0011, SS has 0100, SP has FFFE, BP has 0000, SI has 0000, DI has 0000, DS has 1000, and ES has 0100.
- Memory Dump:** A window titled "Random Access Memory" showing memory starting at address 1000:5000. The dump shows a sequence of bytes: 12 34 56 78 68 AC 00 00-00 00 00 00 00 00 00 00 00 followed by a series of dots. The status bar indicates a size of 140xh5.
- Emulator Window:** Shows the assembly code and the current instruction being executed, which is hlt at address 0100:0011.

For Obj. 2

The screenshot shows a debugger interface with the following components:

- Assembly Editor:** Displays the assembly code for obj.2.asm, identical to obj.1.asm.
- Registers Window:** Shows the state of CPU registers. AX has value 66, BX has 12, CX has 00 00, DX has 00 00, CS has 0100, IP has 0011, SS has 0100, SP has FFFE, BP has 0000, SI has 0000, DI has 0000, DS has 1000, and ES has 0100.
- Memory Dump:** A window titled "Random Access Memory" showing memory starting at address 1000:5000. The dump shows a sequence of bytes: 78 56 12 32 66 24 00 00-00 00 00 00 00 00 00 00 followed by a series of dots. The status bar indicates a size of xU12F5.
- Emulator Window:** Shows the assembly code and the current instruction being executed, which is hlt at address 0100:0011.

For Obj. 3

The screenshot shows the COA Lab 2 software interface. The assembly editor window displays the following code:

```

01 nov ax,1000h
02 nov ds,ax
03 nov ax,[5000h]
04 nov bx,[5002h]
05 nul bx
06 nov [5004h],ax
07 nov [5006h],dx
08 hit
09
10
11
12
13
14
15
16
17
18

```

The memory dump window shows the memory starting at address 1000:5000:

	1000:5000	1000:5020	1000:5030	1000:5040	1000:5050	1000:5060					
1000:5000	12 12 32 21 84 D9 57 02-00 00 00 00 00 00 00 00	1000:5020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1000:5030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1000:5040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1000:5050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	1000:5060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

The emulator window shows the assembly code and registers for obj. 3.bin:

Registers	H	L	Value	Description
AX	D9	84	00 00	MOV AX, 01000h
BX	21	32	00 00	MOV DS, AX
CX	00	00	00 00	MOV AX, [05000h]
DX	02	57	00 00	MOV BX, [05002h]
CS	0100			MUL BX
IP	0015			MUH DX
SS	0100			MOU DS, AX
SP	FFFE			MOU AX, [05004h]
BP	0000			MOU [05006h], DX
SI	0000			INT 3
DI	0000			NOP
DS	1000			NOP
ES	0100			NOP

For Obj. 4

The screenshot shows the COA Lab 2 software interface. The assembly editor window displays the same code as for obj. 3:

```

01 nov ax,1000h
02 nov ds,ax
03 nov ax,[5000h]
04 nov bx,[5002h]
05 div bx
06 nov [5004h],ax
07 nov [5006h],dx
08 hit
09
10
11
12
13
14
15
16
17
18

```

The memory dump window shows the memory starting at address 1000:5000:

	1000:5000	1000:5020	1000:5030	1000:5040	1000:5050	1000:5060					
1000:5000	11 22 11 22 01 00 00-00 00 00 00 00 00 00 00 00	1000:5020	00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00	1000:5030	00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00	1000:5040	00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00	1000:5050	00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00	1000:5060	00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00

The emulator window shows the assembly code and registers for obj. 4.bin:

Registers	H	L	Value	Description
AX	00	01	00 00	MOV AX, 01000h
BX	22	11	00 00	MOV DS, AX
CX	00	00	00 00	MOV AX, [05000h]
DX	00	00	00 00	MOV BX, [05002h]
CS	0100			MUL BX
IP	0015			MUH DX
SS	0100			MOU DS, AX
SP	FFFE			MOU AX, [05004h]
BP	0000			MOU [05006h], DX
SI	0000			INT 3
DI	0000			NOP
DS	1000			NOP
ES	0100			NOP

For Obj. 5

The screenshot shows a debugger interface with the following components:

- Editor Window:** Displays assembly code for object 5. The code includes instructions like MOU AX, 1000H, MOU DS, AX, MOU BX, 13002H, MOU CX, 13004H, LOOP: ADD AX, BX, DEC CX, JNZ LOOP, MOU [13006H], AX, and HLT.
- Registers Window:** Shows the state of CPU registers (AX, BX, CX, DX, CS, IP, SS, SP, BP, SI, DI, DS, ES) across memory locations from 0100:0018 to 0100:0015.
- Memory Dump Window:** A "Random Access Memory" window showing the memory starting at address 1000:3002. It displays a series of NOP (00) bytes followed by some data.
- Emulator Window:** Shows the assembly code again, with the instruction at address 0100:0018 highlighted in blue (MOU AX, 01000h).

For Obj. 6

The screenshot shows a debugger interface with the following components:

- Editor Window:** Displays assembly code for object 6. The code includes instructions like MOU AX, 1000H, MOU DS, AX, MOU BX, 13002H, MOU CX, 0000H, L2: CMP AX, BX, INC CX, SUB AX, BX, INC CX, JMP L2, MOU [3004H], AX, MOU [13006H], CX, and HLT.
- Registers Window:** Shows the state of CPU registers (AX, BX, CX, DX, CS, IP, SS, SP, BP, SI, DI, DS, ES) across memory locations from 0100:001F to 0100:001E.
- Memory Dump Window:** A "Random Access Memory" window showing the memory starting at address 1000:3000. It displays a series of NOP (00) bytes followed by some data.
- Emulator Window:** Shows the assembly code again, with the instruction at address 0100:001F highlighted in blue (CMP AX, BX).

```
01 ;NAME - Yash Rout
02 ;REG- 2041018131
03 ORG 100H
04 MOV AX, 2000H
05 MOV BX, 6200H
06 MOV DX, AX
07 MOV SI, BX
08 MOV CX, AX
09 MOV [1000H], AX
10 OR AX, BX
11 AND [1000H], BX
12 XOR CX, BX
13 NOT DX
14 NOT SI
15 HLT
```

The screenshot shows the Z80 Emulator interface with the title bar "emulator: mycode1.com_". The menu bar includes "file", "math", "debug", "view", "external", "virtual devices", "virtual drive", and "help". Below the menu is a toolbar with icons for "Load" (floppy disk), "reload" (refresh), "step back" (left arrow), "single step" (right arrow), "run" (green triangle), and "step delay ms: 0" (text input). The main window is divided into two panes. The left pane, titled "registers", lists the following register values:

	H	L
AX	62	00
BX	62	00
CX	42	00
DX	DF	FF
CS	0700	
IP	011B	
SS	0700	
SP	FFFE	
BP	0000	
SI	90FF	
DI	0000	
DS	0700	
ES	0700	

The right pane displays assembly code starting at address 0700:011B. The code consists of the following instructions:

```
0700:011B    F4 2441      ; HLT
0700:011C    90 144E      ; NOP
0700:011D    90 144E      ; NOP
0700:011E    90 144E      ; NOP
0700:011F    90 144E      ; NOP
0700:0120    90 144E      ; NOP
0700:0121    90 144E      ; NOP
0700:0122    90 144E      ; NOP
0700:0123    90 144E      ; NOP
0700:0124    90 144E      ; NOP
0700:0125    90 144E      ; NOP
0700:0126    90 144E      ; NOP
0700:0127    90 144E      ; NOP
0700:0128    90 144E      ; NOP
0700:0129    90 144E      ; NOP
0700:012A    90 144E      ; NOP
0700:012B    90 144E      ; NOP
0700:012C    90 144E      ; NOP
```

 original source code

```
01 ;NAME - Yash Rout
02 ;REG- 2041018131
03 ORG 100H
04 MOV AX, 2000H
05 MOV BX, 6200H
06 MOV DX, AX
07 MOV SI, BX
08 MOV CX, AX
09 MOV [1000H], AX
10 OR AX, BX
11 AND [1000H], BX
12 XOR CX, BX
13 NOT DX
14 NOT SI
15 HLT
```

```
01 ;NAME - Yash Rout
02 ;REG- 2041018131
03 ORG 100H
04 MOV AL, 05H
05 NOT AL
06 ADD AL, 01H
07 MOV CL, AL
08 HLT
```

The screenshot shows the emulator interface with the title "emulator: mycode1.com_". The menu bar includes file, math, debug, view, external, virtual devices, virtual drive, and help. The toolbar features icons for Load, reload, step back, single step, run, and step delay ms: 0. The registers window displays the following values:

	H	L
AX	00	FB
BX	00	00
CX	00	FB
DX	00	00
CS	0700	
IP	0108	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

The assembly code window shows the following instructions:

Address	OpCode	OpName	Description
07100	B0	176	
07101	05	005	
07102	F6	246	
07103	D0	208	
07104	04	004	
07105	01	001	
07106	8A	138	
07107	C8	200	
07108	F4	244	
07109	90	144	
0710A	90	144	
0710B	90	144	
0710C	90	144	
0710D	90	144	
0710E	90	144	
0710F	90	144	
07110	90	144	
07111	90	144	
07112	90	144	
07113	90	144	
07114	90	144	
07115	90	144	

The right pane shows the current assembly instruction: HLT.

 original source code

```
01 ;NAME - Yash Rout
02 ;REG- 2041018131
03 ORG 100H
04 MOV AL,05H
05 NOT AL
06 ADD AL,01H
07 MOV CL,AL
08 HLT
09
10
```

```

01 ;NAME - Yash Rout
02 ;REG- 2041018131
03 ORG 100H
04 MOV AL,8FH
05 MOV BL,AL
06 SHR AL,1
07 XOR AL,BL
08 MOV [500H],AL
09 HLT

```

original source code

```

01 ;NAME - Yash Rout
02 ;REG- 2041018131
03 ORG 100H
04 MOV AL,8FH
05 MOV BL,AL
06 SHR AL,1
07 XOR AL,BL
08 MOV [500H],AL
09 HLT
10
11

```

emulator: mycode1.com_

The screenshot shows a debugger window with two panes. The left pane displays assembly code from address 0700:010B to 0700:010B. The right pane shows the CPU register state at address 0700:010B. The registers pane includes fields for H and L, and lists AX, BX, CX, DX, SS, SP, BP, SI, DI, DS, and ES. The CPU pane shows the instruction at 0700:010B is a HLT instruction.

```

file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers H L
AX 00 C8
BX 00 8F
CX 00 0C
DX 00 00
SS 0700
SP FFFE
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
0700:010B: MOU AL, 08Fh
0700:010B: MOU BL, AL
0700:010B: SHR AL, 1
0700:010B: XOR AL, BL
0700:010B: MOU [00500h], AL
0700:010B: HLT
0700:010B: NOP
0700:010B: ...
0700:010B: F4 244 r
0700:010C: 90 144 E
0700:010D: 90 144 E
0700:010E: 90 144 E
0700:010F: 90 144 E
0700:0110: 90 144 E
0700:0111: 90 144 E
0700:0112: 90 144 E
0700:0113: 90 144 E
0700:0114: 90 144 E
0700:0115: 90 144 E
0700:0116: 46 070 F
0700:0117: 88 136 E
0700:0118: 04 004 d
0700:0119: 46 070 F
0700:011A: 88 136 E
0700:011B: 0C 012 q
0700:011C: F4 244 r
0700:011D: 90 144 E
0700:011E: 90 144 E
0700:011F: 90 144 E
0700:0120: 90 144 E
0700:0121: 90 144 E
0700:0122: 90 144 E
0700:0123: 90 144 E
0700:0124: 90 144 E
0700:0125: 90 144 E
0700:0126: 90 144 E
0700:0127: 90 144 E
0700:0128: 90 144 E
0700:0129: 90 144 E
0700:012A: 90 144 E
0700:012B: 90 144 E
0700:010B: screen source reset aux vars debug stack flags

```

new open examples save | compile emulate calculator converter options help about

```

01 ;NAME - Yash Rout
02 ;REG- 2041018131
03
04 ORG 100H
05 MOV SI, 0500H
06 MOV AL, [SI]
07 MOV CL, 04H
08 ROL AL, CL
09 MOV BL, [SI]
10 INC SI
11 MOV CL, [SI]
12 MOV DL, CL
13 AND CL, BL
14 XOR DL, BL
15 OR CL, DL
16 INC SI
17 MOV [SI], AL
18 INC SI
19 MOV [SI], CL
20 HLT

```

original source code

```

05 MOV SI, 0500H
06 MOV AL, [SI]
07 MOV CL, 04H
08 ROL AL, CL
09 MOV BL, [SI]
10 INC SI
11 MOV CL, [SI]
12 MOV DL, CL
13 AND CL, BL
14 XOR DL, BL
15 OR CL, DL
16 INC SI
17 MOV [SI], AL
18 INC SI
19 MOV [SI], CL
20 HLT

```

emulator: mycode1.com_

The screenshot shows a debugger window with two panes. The left pane displays assembly code from address 0700:011C to 0700:011C. The right pane shows the CPU register state at address 0700:011C. The registers pane includes fields for H and L, and lists AX, BX, CX, DX, SS, SP, BP, SI, DI, DS, and ES. The CPU pane shows the instruction at 0700:011C is a HLT instruction.

```

file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers H L
AX 00 00
BX 00 00
CX 00 00
DX 00 00
SS 0700
SP FFFE
BP 0000
SI 0503
DI 0000
DS 0700
ES 0700
0700:011C: MOU SI, 00500h
0700:011C: MOU AL, [SI]
0700:011C: MOV CL, 04h
0700:011C: ROL AL, CL
0700:011C: MOU BL, [SI]
0700:011C: INC SI
0700:011C: MOU CL, [SI]
0700:011C: MOU DL, CL
0700:011C: AND CL, BL
0700:011C: XOR DL, BL
0700:011C: OR CL, DL
0700:011C: INC SI
0700:011C: MOV [SI], AL
0700:011C: INC SI
0700:011C: MOU [SI], CL
0700:011C: HLT
0700:011C: NOP
0700:011C: NOP
0700:011C: NOP
0700:011C: NOP
0700:011C: ...
0700:011C: F4 244 r
0700:011D: 90 144 E
0700:011E: 90 144 E
0700:011F: 90 144 E
0700:0120: 90 144 E
0700:0121: 90 144 E
0700:0122: 90 144 E
0700:0123: 90 144 E
0700:0124: 90 144 E
0700:0125: 90 144 E
0700:0126: 90 144 E
0700:0127: 90 144 E
0700:0128: 90 144 E
0700:0129: 90 144 E
0700:012A: 90 144 E
0700:012B: 90 144 E
0700:011C: screen source reset aux vars debug stack flags

```

The screenshot shows the COA LAB 1 emulator interface. The top menu bar includes file, edit, bookmarks, assembler, emulator, math, ascii codes, and help. The main window displays assembly code on the left and a debugger interface on the right.

Assembly Code:

```
01 ;NAME - Yash Rout
02 ;REG- 2041018131
03 ;CSIT-D
04 mov si, 1000h
05 mov cl, [si]
06 mov ch, 00h
07 mov bx, cx
08 mov ax, 0000h
09 12: inc si
10 inc si
11 add ax, [si]
12 jnc 11
13 inc ch
14 l1: dec cl
15 jnz 12
16 inc si
17 inc si
18 mov [si], ax
19 inc si
20 inc si
21 mov [si], ch
22 div bx
23 inc si
24 inc si
25 mov [si], ax
26 inc si
27 inc si
28 mov [si], dx
29 hlt
```

Registers:

	H	L
AX	00	00
BX	00	00
CX	00	00
DX	00	00
CS	F400	
IP	0174	
SS	0100	
SP	FFF8	
BP	0000	
SI	1204	
DI	0000	
DS	0100	
ES	0100	

Memory Dump:

	F400:0174	F400:0174
F4170:	FF 255 RES	BIOS DI
F4171:	FF 255 RES	INT 00h
F4172:	CD 205 =	IRET
F4173:	00 000 NULL	ADD [BX + SI], AL
F4174:	CF 207 ±	ADD [BX + SI], AL
F4175:	00 000 NULL	ADD [BX + SI], AL
F4176:	00 000 NULL	ADD [BX + SI], AL
F4177:	00 000 NULL	ADD [BX + SI], AL
F4178:	00 000 NULL	ADD [BX + SI], AL
F4179:	00 000 NULL	ADD BH, BH
F417A:	00 000 NULL	DEC BP
F417B:	00 000 NULL	ADD AL, 0CFh
F417C:	00 000 NULL	ADD [BX + SI], AL
F417D:	00 000 NULL	ADD [BX + SI], AL
F417E:	00 000 NULL	ADD [BX + SI], AL
F417F:	00 000 NULL	ADD [BX + SI], AL
F4180:	FF 255 RES	ADD BH, BH
F4181:	FF 255 RES	DEC BP
F4182:	CD 205 =	ADC BH, CL
F4183:	04 004 ♦	ADD [BX + SI], AL
F4184:	CF 207 ±	ADD [BX + SI], AL
F4185:	00 000 NULL	...

Control Buttons:

- Load
- reload
- step back
- single step
- run
- step delay ms: 0

Source Code View:

```
10 inc si
11 add ax, [si]
12 jnc 11
13 inc ch
14 l1: dec cl
15 jnz 12
16 inc si
17 inc si
18 mov [si], ax
19 inc si
20 inc si
21 mov [si], ch
22 div bx
23 inc si
24 inc si
25 mov [si], ax
```

The screenshot shows a debugger interface with the following details:

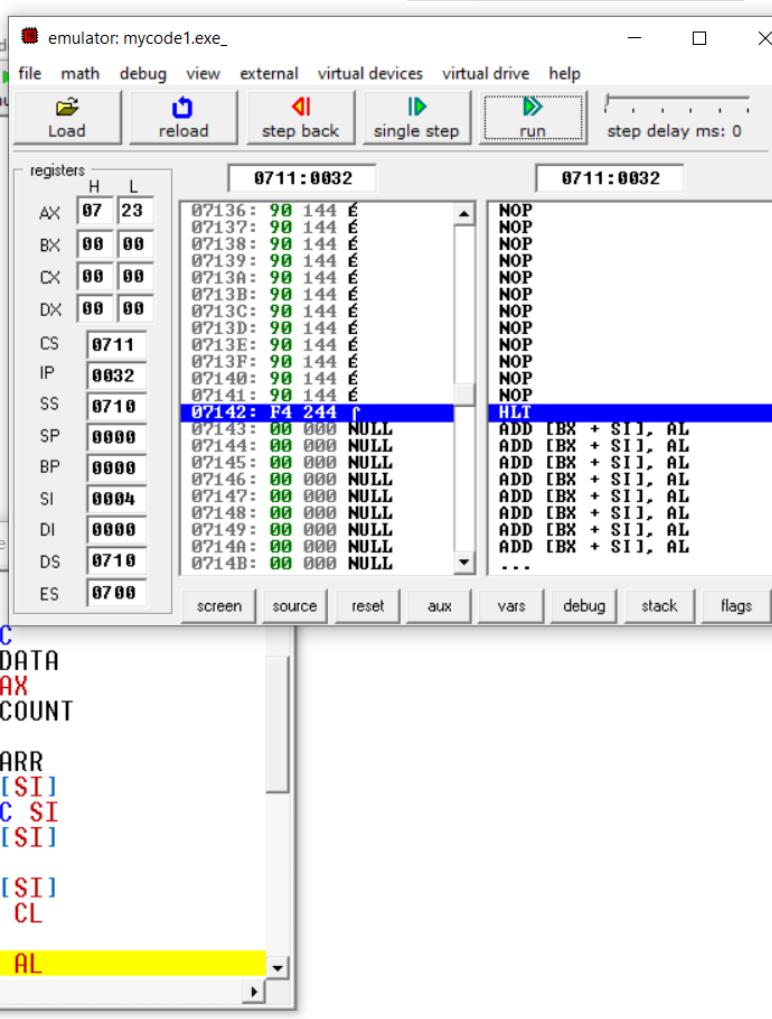
- File Menu:** new, open, examples, save, compile, emulate.
- Title Bar:** emulator: mycode1.bin
- Toolbar:** Load, reload, step back, single step, run, step delay ms: 0
- Registers Window:** Shows CPU registers (SI, AL, CL, BL) with values: SI = 01000h, AL = 00h, CL = 08h, BL = 00h.
- Memory Dump Window:** Address 0100:0016 to 0100:0014. Content:

0100:0016	.6: F4 244
0100:0017	? 90 144
0100:0018	? 90 144
0100:0019	? 90 144
0100:001A	? 90 144
0100:001B	? 90 144
0100:001C	? 90 144
0100:001D	? 90 144
0100:001E	? 90 144
0100:001F	? 90 144
0100:0020	? 90 144
0100:0021	? 90 144
0100:0022	? 90 144
0100:0023	? 90 144
0100:0024	? 90 144
0100:0025	? 90 144
0100:0026	? 90 144
0100:0027	? 90 144
0100:0028	? 90 144
0100:0029	? 90 144
0100:002A	? 90 144
0100:002B	? F4 244
- Source Code Window:** Shows the assembly source code with labels loop1 and loop2. The instruction at address 0100:0016 is highlighted in yellow.
- Bottom Navigation:** source, reset, aux, vars, debug, stack, flags.

```

file edit bookmarks assembler emulator math ascii code
new open examples save compile emulate
01 ;NAME - Yash Rout
02 ;REG- 2041018131
03 ;CSIT- D
04 .DATA
05
06 COUNT DB 04H
07 ARR DB 09H, 04H, 23H, 05H
08 RES DB ?
09 .CODE
10 MAIN PROC
11 MOV AX, DATA
12 MOV DS, AX
13 MOV CL, COUNT
14 DEC CL
15 LEA SI, ARR
16 MOV AL, [SI]
17 BACK: INC SI
18 CMP AL, [SI]
19 JNC FWD
20 MOV AL, [SI]
21 FWD: DEC CL
22 JNZ BACK
23 MOV RES, AL
24 END MAIN

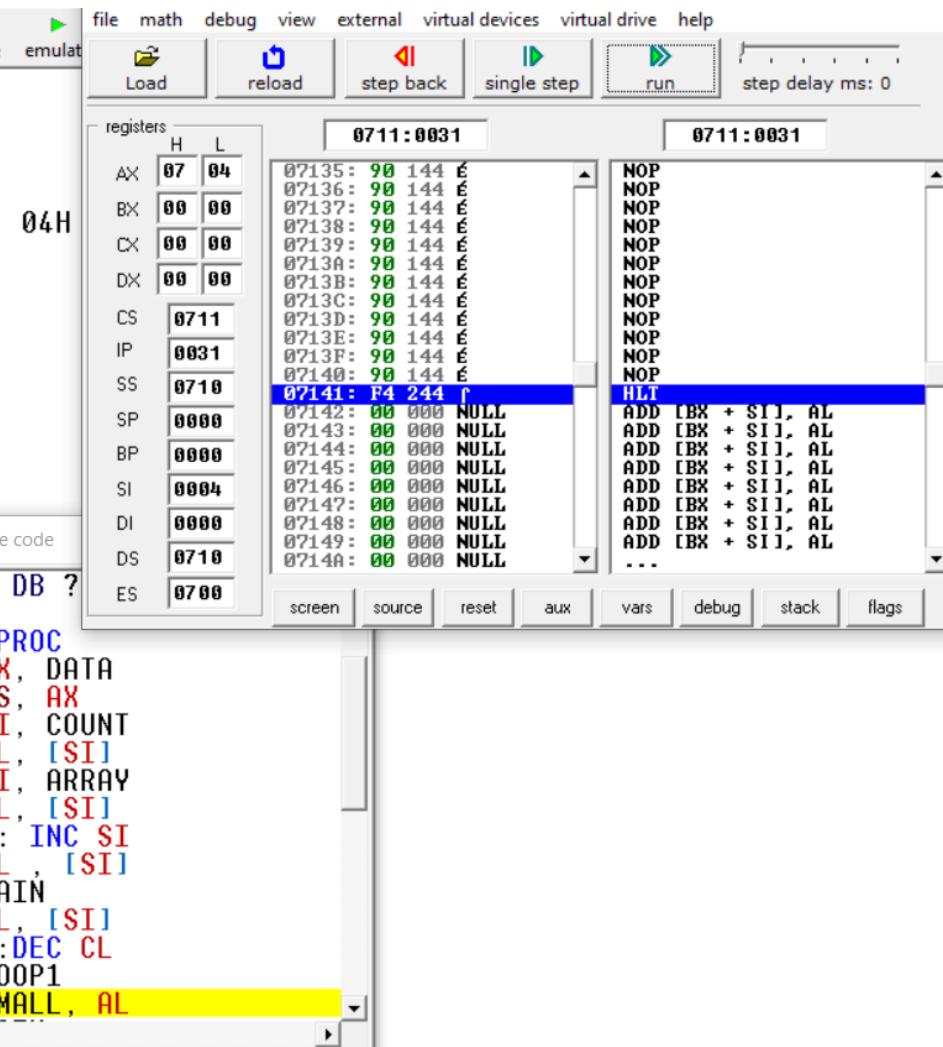
```



```

file math debug view external virtual devices virtual drive help
new open examples save compile emulate
01 ;NAME - Yash Rout
02 ;REG- 2041018131
03 ;CSIT- D
04 .DATA
05 COUNT DB 03H
06 ARRAY DB 09H, 05H, 20H, 04H
07 SMALL DB ?
08 .CODE
09 MAIN PROC
10 MOV AX, DATA
11 MOV DS, AX
12 LEA SI, COUNT
13 MOV CL, [SI]
14 LEA SI, ARRAY
15 MOV AL, [SI]
16 LOOP1: INC SI
17 CMP AL, [SI]
18 JS AGAIN
19 MOV AL, [SI]
20 AGAIN: DEC CL
21 JNZ LOOP1
22 MOV SMALL, AL
23 END MAIN

```



new open examples save file math debug view external virtual devices virtual drive help

compile emulate cal Load reload step back single step run step delay ms: 0

```

01 ;NAME - Yash Rout
02 ;REG- 2041018131
03 ;CSIT- D
04 ;EXP 5 OBJ 3
05
06 .DATA
07 COUNT DB 04H
08 ARR DB 0F4H, 32H, 05H, 04H
09 .CODE
10 MAIN PROC
11 MOV AX, DATA
12 MOV DS, AX
13 MOV CH, COUNT
14 DEC CH
15 UP2: MOV CL, CH
16 LEA SI, ARR
17 UP1: MOV AL, [SI]
18 CMP AL, [SI+1]
19 JC DOWN
20 MOV DL, [SI+1]
21 XCHG [SI], DL
22 MOV [SI+1], DL
23 DOWN: INC SI
24 DEC CL
25 JNZ UP1
26 DEC CH
27 JNZ UP2
28
29 END MAIN

```

original source code

	H	L	Address	OpCode	Mnemonic	Description
AX	07	05	07148:	90	144	É
BX	00	00	07149:	90	144	É
CX	00	00	0714A:	90	144	É
DX	00	05	0714B:	90	144	É
CS	0711		0714C:	F4	244	↑
IP	003C		0714D:	00	000	NULL
SS	0710		0714E:	00	000	NULL
SP	0000		0714F:	00	000	NULL
BP	0000		07150:	00	000	NULL
SI	0002		07151:	00	000	NULL
DI	0000		07152:	00	000	NULL
DS	0710		07153:	00	000	NULL
ES	0700		07154:	00	000	NULL
			07155:	00	000	NULL
			07156:	00	000	NULL
			07157:	00	000	NULL
			07158:	00	000	NULL
			07159:	00	000	NULL
			0715A:	00	000	NULL
			0715B:	00	000	NULL
			0715C:	00	000	NULL
			0715D:	00	000	NULL

screen source reset aux vars debug stack flags

new open examples save file math debug view external virtual devices virtual drive help

compile emulate cal Load reload step back single step run step delay ms: 0

```

01 ;NAME - Yash Rout
02 ;REG- 2041018131
03 ;CSIT- D
04 ;LAB 5 OBJ 4
05
06 .DATA
07 COUNT DB 04H
08 ARRAY DB 0F4H, 32H, 05H, 04H
09 .CODE
10 MAIN PROC
11 MOV AX, DATA
12 MOV DS, AX
13 MOV CH, COUNT
14 DEC CH
15 UP2: MOV CL, CH
16 LEA SI, ARRAY
17 UP1: MOV AL, [SI]
18 CMP AL, [SI+1]
19 JNC DOWN
20 MOV DL, [SI+1]
21 DOWN: INC SI
22 DEC CL
23 JNZ UP1
24 DEC CH
25 JNZ UP2
26 END MAIN

```

original source code

	H	L	Address	OpCode	Mnemonic	Description
AX	07	04	07147:	F4	244	↑
BX	00	00	07148:	00	000	NULL
CX	00	00	07149:	00	000	NULL
DX	00	32	0714A:	00	000	NULL
CS	0711		0714B:	00	000	NULL
IP	0037		0714C:	00	000	NULL
SS	0710		0714D:	00	000	NULL
SP	0000		0714E:	00	000	NULL
BP	0000		0714F:	00	000	NULL
SI	0002		07150:	00	000	NULL
DI	0000		07151:	00	000	NULL
DS	0710		07152:	00	000	NULL
ES	0700		07153:	00	000	NULL
			07154:	00	000	NULL
			07155:	00	000	NULL
			07156:	00	000	NULL
			07157:	00	000	NULL
			07158:	00	000	NULL
			07159:	00	000	NULL
			0715A:	00	000	NULL
			0715B:	00	000	NULL
			0715C:	00	000	NULL

screen source reset aux vars debug stack flags

Observation (with screenshots)

Objective 1

The screenshot shows the CPULator ARMv7 System Simulator interface. The assembly code in the editor is:

```
1 @Name-Sumit Pradhan
2 @2041016245
3 @CSIT-A
4
5 @EXP 6 -> OBJ 1
6 .global _start
7 _start:
8 LDR R0,=LIST
9 LDR R1,[R0]
10 LDR R2,[R0,#4]
11 ADD R3,R1,R2
12
13 END:
14 B END
15
16 .data
17 LIST:
18 .WORD 0X12,0X11
19
20
```

The Registers window shows the following values:

r0	00000018
r1	00000012
r2	00000011
r3	00000023
r4	00000000
r5	00000000
r6	00000000
r7	00000000
r8	00000000
r9	00000000
r10	00000000
r11	00000000
r12	00000000
sp	00000000
lr	00000000

The Messages window shows the compilation process:

```
Compiling...
Code and data loaded from ELF executable into memory. Total size is 32 bytes.
Assemble: arm-altera-eabi-as -mfloat-abi=soft -march=armv7-a -mcpu=cortex-a9 -mfpu=neon-fp16 --gdwarf2 -o work/asmBfNkD4.s.o work/asmBfNkD4.s
Link: arm-altera-eabi-ld --script build_arm.ld -e _start -u _start -o work/asmBfNkD4.self work/asmBfNkD4.s.o
Compile succeeded.
```

Objective 2

The screenshot shows the CPULator ARMv7 System Simulator interface. The assembly code in the editor is identical to Objective 1:

```
1 @Name-Sumit Pradhan
2 @2041016245
3 @CSIT-A
4
5 @EXP 6 -> OBJ 2
6 .global _start
7 _start:
8 LDR R0,=LIST
9 LDR R1,[R0]
10 LDR R2,[R0,#4]
11 SUBS R3,R1,R2
12
13 END:
14 B END
15
16 .data
17 LIST:
18 .WORD 0X12,0X11
19
20
```

The Registers window shows the following values:

r0	00000018
r1	00000012
r2	00000011
r3	00000001
r4	00000000
r5	00000000
r6	00000000
r7	00000000
r8	00000000
r9	00000000
r10	00000000
r11	00000000
r12	00000000
sp	00000000
lr	00000000

The Messages window shows the compilation process:

```
Compiling...
Code and data loaded from ELF executable into memory. Total size is 32 bytes.
Assemble: arm-altera-eabi-as -mfloat-abi=soft -march=armv7-a -mcpu=cortex-a9 -mfpu=neon-fp16 --gdwarf2 -o work/asm6sLg1H.s.o work/asm6sLg1H.s
Link: arm-altera-eabi-ld --script build_arm.ld -e _start -u _start -o work/asm6sLg1H.self work/asm6sLg1H.s.o
Compile succeeded.
```

Objective 3

The screenshot shows the CPULATOR ARMv7 System Simulator interface. The assembly code in the editor is:

```
1 @Name=Sumit Pradhan
2 @2041016245
3 @CSIT-A
4
5 @EXP 6 -> OBJ 3
6 .global _start
7 _start:
8 LDR R0,=LIST
9 LDR R1,[R0]
10 LDR R2,[R0,#4]
11 MUL R3,R1,R2
12
13 END:
14 B END

16 .data
LIST:
18 .WORD 0X12,0X10
19
20
21
22
```

The Registers window shows the following values:

Register	Value
r0	00000018
r1	00000012
r2	00000010
r3	00000128
r4	00000000
r5	00000000
r6	00000000
r7	00000000
r8	00000000
r9	00000000
r10	00000000
r11	00000000
r12	00000000
sp	00000000
lr	00000000

The Messages window shows the compilation process:

```
Compiling...
Code and data loaded from ELF executable into memory. Total size is 32 bytes.
Assemble: arm-altera-eabi-as -mfloat-abi=soft -march=arm7-a -mcpu=cortex-a9 -mfpu=neon-fp16 --gdwarf2 -o work/asmrReL4w.s.o work/asmrReL4w.s
Link: arm-altera-eabi-ld --script build_arm.ld -e _start -u _start -o work/asmrReL4w.self work/asmrReL4w.s.o
Compile succeeded.
```

Objective 4

The screenshot shows the CPULATOR ARMv7 System Simulator interface. The assembly code in the editor is:

```
1 @Name=Sumit Pradhan
2 @2041016245
3 @CSIT-A
4
5 @EXP 6 -> OBJ 4
6 .global _start
7 _start:
8 LDR R0,=LIST
9 LDR R1,[R0]
10 LDR R2,[R0,#4]
11 AND R3,R2,R1
12 ORR R4,R2,R1
13 EOR R5,R2,R1
14 MVN R6,R2
15
16 END:
17 B END

19 .data
LIST:
20 .WORD 0X40,0X41
21
22
```

The Registers window shows the following values:

Register	Value
r0	00000028
r1	00000040
r2	00000041
r3	00000040
r4	00000041
r5	00000001
r6	fffffbff
r7	00000000
r8	00000000
r9	00000000
r10	00000000
r11	00000000
r12	00000000
sp	00000000
lr	00000000

The Messages window shows the compilation process:

```
Compiling...
Code and data loaded from ELF executable into memory. Total size is 48 bytes.
Assemble: arm-altera-eabi-as -mfloat-abi=soft -march=arm7-a -mcpu=cortex-a9 -mfpu=neon-fp16 --gdwarf2 -o work/asmcCafsE.s.o work/asmcCafsE.s
Link: arm-altera-eabi-ld --script build_arm.ld -e _start -u _start -o work/asmcCafsE.self work/asmcCafsE.s.o
Compile succeeded.
```

Observation (with screenshots)

Objective 1

The screenshot shows a debugger interface with several panes:

- Registers**: Shows registers r0 to r15, all set to 0. A note indicates a value of 0x000001d3 is present at address 00000000.
- Disassembly (Ctrl-D)**: Displays assembly code for a program named "untitled.s". The code includes sections for .start, .back, .fwd, and .exit, along with various memory operations like ldr, mov, and cmp.
- Editor (Ctrl-E)**: Shows the source code for the program. It includes comments for lab assignments and defines symbols like _start and _array.
- Messages**: Displays build logs for arm-altera-eabi-as and arm-altera-eabi-ld.
- Interrupts**: Shows three Cortex-A9 Watchdog Timer entries in the interrupt list.

Objective 2

The screenshot shows a debugger interface with several panes:

- Registers**: Shows registers r0 to r15, all set to 0. A note indicates a value of 0x000001d3 is present at address 00000000.
- Disassembly (Ctrl-D)**: Displays assembly code for a program named "untitled.s". The code includes sections for .start, .back, .fwd, and .exit, along with various memory operations like ldr, mov, and cmp.
- Editor (Ctrl-E)**: Shows the source code for the program. It includes comments for lab assignments and defines symbols like _start and _array.
- Messages**: Displays build logs for arm-altera-eabi-as and arm-altera-eabi-ld. A warning message about the entry symbol _start is shown.
- Interrupts**: Shows three Cortex-A9 Watchdog Timer entries in the interrupt list.

Objective 3

Stopped Step Into F2 Step Over Ctrl+F2 Step Out Shift+F2 Continue F3 Stop F4 Restart Ctrl+Shift+R Reload Ctrl+Shift+L File Help

Registers Refresh Address Opcode Disassembly Go to address, label, or register: 00000000

Address	Opcode	Disassembly
fffffe8	aaaaaaaa	bge 0xfeaaaa98
fffffec	aaaaaaaa	bge 0xfeaaaa9c
ffffff0	aaaaaaaa	bge 0xfeaaaaa0
ffffff4	aaaaaaaa	bge 0xfeaaaaa4
fffffb8	aaaaaaaa	bge 0xfeaaaaa8
ffffffc	aaaaaaaa	bge 0xfeaaaaac
1		@Harsh Kumar
2		@2641011125
3		@Lab7 Obj3 - EVEN OR ODD number from a given array
4		.global _start
5		_start:
6		ldr r8, =count
7		ldr r1, [r8]
8		ldr r3, =array
9		ldr r3, [pc, #44] ; 0x3c
10		ldr r4,=even
11		ldr r4, [pc, #44] ; 0x40
12		ldr r5, =odd
13		ldr r5, [pc, #44] ; 0x44
14		back:
15		ldr r6, [r3], #4
16		ldr r6, [r6]
17		ldr r3, [r6]
18		ldr r3, [pc, #44] ; 0x38
19		beq fwd
20		str r6,[r5],#4
21		b fwd1
22		fwd: str r6,[r4],#4
23		fwd1: subs r1,r1,#01
24		bne back
25		exit: b exit
26		.data
27		count:.word 0x07
28		array:.word 0x15,0x35,0x32,0x45,0x10,0x4f,0x34
29		even:.word 0,0,0
30		odd:.word 0,0,0
31		Result:
32		

Editor (Ctrl+E) Compile and Load (F5) Language: ARMv7 untitled.s [changed since save]

Registers Call stack Trace Breakpoints Watchpoints Symbols Counters Settings Number Display Options Size: Word Format: Hexadecimal Memory words per row: 4

Messages Code and data loaded from ELF executable into memory. Total size is 136 bytes.

```
Assemble: arm-altera-eabi-as -mfloat-abt=soft -march=armv7-a -mcpu=cortex-a9 -mfpu=neon-fp16 -gdwarf2 -o work/armHwIvuW.s.o work/armHwIvuW.o
Link: arm-altera-eabi-ld --script build_arm.ld -e _start -u _start -o work/armHwIvuW.s.elf work/armHwIvuW.o
Compile succeeded.
```

Disassembly (Ctrl-D) Memory (Ctrl-M) Clear Cortex-A9 Watchdog Timer IRQ 30 fffec620 Once Stop TO=0 RST=0
 HPS L4 Watchdog Timer IRQ 203 ffdb2068 2147483648 Interrupt Stop INTR=0
 HPS L4 Watchdog Timer IRQ 204 ffde3000 2147483648 Interrupt Stop INTR=0