Topic: 1.4.4 Assembly Language

May/June 2012. P31/32

5 The table shows the assembly language instructions for a processor which has one general purpose register – the Accumulator.

Instruction		Explanation
Op Code	Operand	
LDD	<address>	Load using direct addressing
STO	<address>	Store the contents of the Accumulator at the given address
LDI	<address>	Load using indirect addressing
LDX	<address>	Load using indexed addressing
INC		Add 1 to the contents of the Accumulator
END		End the program and return to the operating system

(c) Trace this assembly language program using the trace table below.

```
500     LDD  507
501     INC
502     STO  509
503     LDD  508
504     INC
505     STO  510
506     END
507      22
508     170
509       0
510       0
```

Accumulator	Memory Address			
	507	508	509	510
	22	170	0	0

[5]

Topic: 1.4.4 Assembly Language

May/June 2012. P33

6 The table shows the assembly language instructions for a processor which has one general purpose register – the Accumulator.

Instruction		Explanation
Op Code	Operand	
LDD	<address>	Load using direct addressing
STO	<address>	Store the contents of the Accumulator at the given address
LDI	<address>	Load using indirect addressing
LDX	<address>	Load using indexed addressing
INC		Add 1 to the contents of the Accumulator
END		End the program and return to the operating system

(c) Trace this assembly language program using the given trace table. The first instruction of the program is loaded into main memory at address 200.

```
200    LDD  208
201    INC
202    STO  208
203    LDD  207
204    INC
205    STO  207
206    END
207     16
208    150
```

Accumulator	Memory Address	
	207	208
	16	150

[4]

Topic: 1.4.4 Assembly Language

May/June 2013. P31/32

3 The table shows the assembly language instructions for a processor which has one general purpose register – the Accumulator (ACC), and an index register (IX).

Instruction		Explanation
Op Code	Operand	
LDD	<address>	Direct addressing. Load the contents of the given address to ACC
STO	<address>	Store the contents of ACC at the given address
LDI	<address>	Indirect addressing. At the given address is the address to be used. Load the contents of this second address to ACC
LDX	<address>	Indexed addressing. Form the address as <address> + the contents of IX. Copy the contents of this address to ACC
INC	<register>	Add 1 to the contents of the register (ACC or IX)
ADD	<address>	Add the contents of the given address to the contents of ACC
OUT		Output the contents of ACC (as a denary number) to the monitor
IN		Input a denary number from the keyboard and store in ACC
END		End the program and return to the operating system

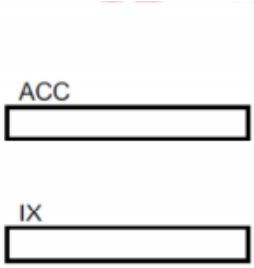
The diagram shows a program loaded in main memory starting at location 100.

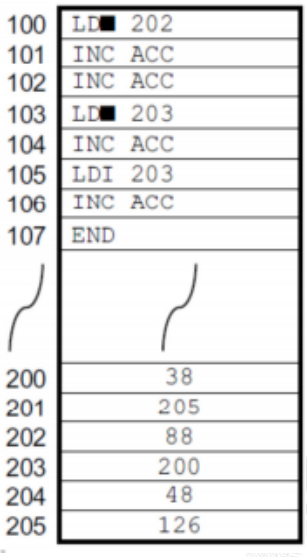
Two of the op-codes have been partially blanked out.

Locations 200 onwards contain data which is used by the program.

(c) The instruction at address 105 is fetched and executed.

Draw on the diagram to explain how this instruction is executed and show the contents of ACC after execution.





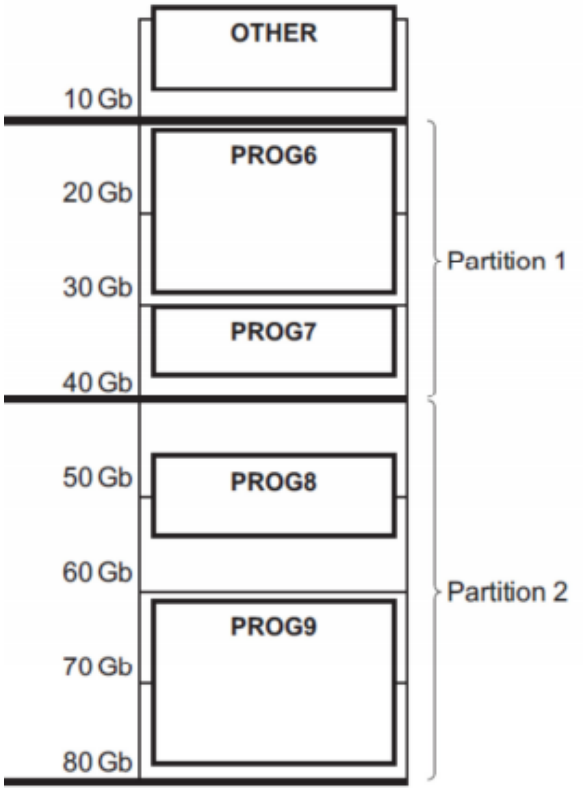
[2]

May/June 2013. P31/32

6 A multiprogramming, multi-user operating system organises the available memory into two fixed sized partitions.

• Partition 1 – size 30 Gb – is used only for batch processing

• Partition 2 – size 40 Gb – is used only for interactive processing at a terminal

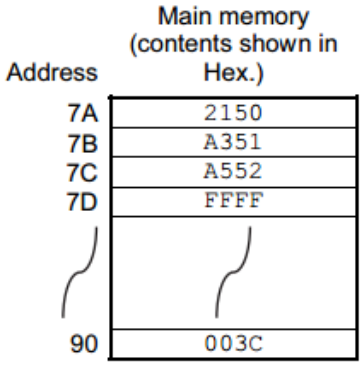


(i) If PROG6 completes execution, which programs (if any) can be loaded next? **[1]**

(ii) If PROG8 completes execution, which programs (if any) can be loaded next? **[1]**

Oct/Nov 2013. P31

3 (c) The diagram shows a program loaded into main memory starting at memory address 7A Hex.



(i) How many bits are used for each main memory location? **[1]**

The trace table below is used to show how the contents of the special-purpose registers change as the program is executed. The steps in the fetch stage of the fetch-execute cycle are shown in the first column using register transfer notation. (For example, `MAR ← [PC]` means the content of the Program Counter is copied to the Memory Address Register.)

(ii) Complete the trace table for the fetching of the **first program instruction (2150):**

- Show the changing contents of the registers
- Put a tick in the Address bus/Data bus column to show when the signals on that bus change.

Fetch stage	Special purpose registers (Contents shown in Hex.)				Buses	
	PC	MAR	MDR	CIR	Address bus	Data bus
	7A					
MAR ← [PC]						
PC ← [PC] + 1						
MDR ← [[MAR]]						
CIR ← [MDR]						

[5]

(d) The following table shows some of a processor's instruction set in assembly language.

Instruction		Explanation
Op Code	Operand	
LDD	<address>	Direct addressing. Load the contents of the given address to ACC
LDI	<address>	Indirect addressing. At the given address is the address to be used. Load the contents of this second address to ACC
LIX	<address>	Load the contents of the address to the Index register (IX)
LDX	<address>	Indexed addressing. Form the address as <address> + the contents of IX. Copy the contents of this address to ACC

The following program is to be executed. Shown are:

- the first four instructions only of this program
- the memory locations which are accessed by this program.

Address	Main memory
100	LIX 200
101	LDD 201
102	LDI 201
103	LDX 201
200	3
201	216
202	99
203	217
204	63
216	96
217	97

Topic: 1.4.4 Assembly Language

Complete the trace table below for the first **four** program instructions. Show each change in the contents of the registers.

Instruction	Register	
	Accumulator (ACC)	Index Register (IX)
LIX 200		
LDD 201		
LDI 201		
LDX 201		

[4]

Oct/Nov 2013. P32

3 (c) The diagram shows a program loaded into main memory starting at memory address 30 Hex.

Address	Main memory (contents shown in Hex.)
30	2150
31	A351
32	A552
33	FFFF
⋮	⋮
58	003C
59	103C
5A	010B

(i) How many bytes are used to store each program instruction? **[1]**

(ii) Describe the steps in the fetch stage of the fetch-execute cycle. Refer to the instruction at address 30 to illustrate your answer. **[5]**

Topic: 1.4.4 Assembly Language

(d) The following table shows some of a processor's instruction set in assembly language.

Instruction		Explanation
Op Code	Operand	
LIX	<address>	Load the contents of the address to the Index register (IX)
LDX	<address>	Indexed addressing. Form the address as <address> + the contents of IX. Copy the contents of this address to ACC
STO	<address>	Store the contents of ACC at the given address
ADD	<address>	Add the contents of the given address to the ACC
INC	<register>	Add 1 to the contents of the register (ACC or IX)
JMP	<address>	Jump to the given address

The following program is to be executed. Shown are:

- the first six instructions of this program
- the memory locations which will be accessed by this program.

Address	Main memory contents
100	LIX 200
101	LDX 200
102	ADD 204
103	STO 204
104	INC IX
105	JMP 101
200	1
201	13
202	14
203	22
204	0

Complete the trace table below for **three** iterations of the loop. Show each change to the contents of the registers and memory location 204.

ACC	IX	Main memory address 204
		0

[4]

Oct/Nov 2013. P33

(c) The diagram shows a program loaded into main memory starting at memory address 40 Hex.

Address	Main memory (Contents shown in Hex.)
40	7324
41	A351
42	A552
43	FFFF
⌇	⌇
68	003C
69	103C
6A	010B

(i) How many bytes are used to store each program instruction? **[1]**

(ii) Describe the steps in the fetch stage of the fetch-execute cycle. Use the instruction at address 40 to illustrate your answer. **[5]**

(d) The following table shows some of a processor's instruction set in assembly language.

Instruction		Explanation
Op Code	Operand	
LDD	<address>	Direct addressing. Load the contents of the given address to ACC
LDI	<address>	Indirect addressing. At the given address is the address to be used. Load the contents of this second address to ACC
STO	<address>	Store the contents of ACC at the given address
ADD	<address>	Add the contents of the given address to the ACC
INC	<register>	Add 1 to the contents of the register (ACC or IX)
JMP	<address>	Jump to the given address

The following program is to be executed. Shown are:

- the first seven instructions in this program
- the memory locations which will be accessed by this program

Address	Main memory
130	LDI 160
131	ADD 153
132	STO 153
133	LDD 160
134	INC ACC
135	STO 160
136	JMP 130
150	13
151	23
152	11
153	0
160	150

Topic: 1.4.4 Assembly Language

Complete the trace table below for **two** iterations of the loop. Show each change in the contents of the register and memory locations.

Register	Memory location	
ACC	153	160
	0	150

[4]

Topic: 1.4.4 Assembly Language

Oct/Nov 2014. P31

Oct/Nov 2014. P33

4 The table below gives a subset of the assembly language instructions for a computer with a single general-purpose register, the Accumulator (ACC), and an index register (IX).

Instruction			Explanation
Opcode (mnemonic)	Operand	Opcode (binary)	
LDD	<address>	0000 0100	Direct addressing. Load the contents of the given address to ACC
LDV	<number>	0000 0101	Load the given number to ACC
STO	<address>	0001 0000	Store the contents of ACC at the given address
LDI	<address>	0000 0110	Indirect addressing. At the given address is the address to be used. Load the contents of this second address to ACC
LDX	<address>	0000 0111	Indexed addressing. Form the address as <address> + the contents of IX. Copy the contents of this address to ACC
INC	<register>	0000 0011	Add 1 to the contents of the register (ACC or IX)
OUTCH		1000 0001	Output to the monitor the character corresponding to the ASCII character code in ACC
IN		1001 0000	Input a denary number from the keyboard and store in ACC
JMP	<address>	1100 1000	Unconditional jump to the given address
END		1111 1111	End the program and return to the operating system

The diagram shows a program loaded in main memory starting at location 100. Locations 200 onwards contain data which are used by the program.

(a) (i) The instruction at address 102 is fetched.

ACC

IX

Show the contents of the registers after execution.

Write on the diagram to explain. **[2]**

100	LDI 150
101	OUTCH
102	LDD 203
103	INC ACC
104	STO 150
105	JP 100
106	END
107	
150	200
200	65
201	76
202	65
203	77
204	32
205	32

(ii) The instruction at address 100 is fetched.

ACC

IX

Show the contents of the registers after execution. Write on the diagram to explain. **[3]**

(b) The given table of instructions shows the binary number used for each instruction's opcode.
All instructions in machine code are stored as a 16-bit pattern, with the opcode as the first 8 bits and the operand as the second 8 bits.

(i) What is the maximum number of different instructions this processor could have? **[1]**

(ii) Consider the instruction:

0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1

Describe what this instruction does. **[2]**

(iii) Programmers prefer to write machine code instructions in hexadecimal. Explain why. **[1]**

(iv) What is the hexadecimal number for the instruction shown in part (b)(ii)? **[1]**

Show the machine code for the following instructions:

(v) `LDI 150`

[2]

(vi) LDV 15

[2]

(c) Use the ASCII code table to trace the **first four iterations** of the given program.

ASCII code table (part)					
Character	**Decimal**	**Character**	**Decimal**	**Character**	**Decimal**
<Space>	32	I	73	R	82
A	65	J	74	S	83
B	66	K	75	T	84
C	67	L	76	U	85
D	68	M	77	V	86
E	69	N	78	W	87
F	70	O	79	X	88
G	71	P	80	Y	89
H	72	Q	81	Z	90

ACC	Location 150	OUTPUT

100	LDI 150
101	OUTCH
102	LDD 150
103	INC ACC
104	STO 150
105	JP 100
106	END
107	
	//
150	200
	//
200	65
201	76
202	65
203	77
204	32
205	32

[5]

Oct/Nov 2014. P32

4 The table below gives a subset of the assembly language instructions for a computer with a single general purpose register, the Accumulator (ACC), and an index register (IX).

Instruction			Explanation
Opcode (mnemonic)	Operand	Opcode (binary)	
LDD	<address>	0000 0100	Direct addressing. Load the contents of the given address to ACC
LDV	<number>	0000 0101	Load the given number to ACC
STO	<address>	0001 0000	Store the contents of ACC at the given address
LDI	<address>	0000 0110	Indirect addressing. At the given address is the address to be used. Load the contents of this second address to ACC
LDX	<address>	0000 0111	Indexed addressing. Form the address as <address> + the contents of IX. Copy the contents of this address to ACC
INC	<register>	0000 0011	Add 1 to the contents of the register (ACC or IX)
OUTCH		1000 0001	Output the character corresponding to the ASCII character code in ACC to the monitor
IN		1001 0000	Input a denary number from the keyboard and store in ACC
JMP	<address>	1100 1000	Unconditional jump to the given address
CMP	<number>	1100 1001	Compare the contents of ACC with the given number
JPE	<address>	1110 0111	If the result of the previous compare instruction was a match, jump to the given address

(a) The given table of instructions shows the binary number used for each instruction's opcode.
All instructions in machine code are stored as a 16-bit pattern, with the opcode as the first 8 bits and the operand as the second 8 bits.

(i) What is the maximum number of memory locations which can be directly addressed? **[1]**

(ii) Consider the instruction:

0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	1

Describe what this instruction does. **[2]**

(iii) Programmers prefer to write machine code instructions in hexadecimal. Explain why. **[1]**

(iv) What is the hexadecimal number for the machine code instruction shown in part (b)(ii)? **[1]**

(v) Show the 16-bit machine code for the following instruction:

```
JPE 204
```


[2]

(b) Use the ASCII code table to trace execution of the given program.

ASCII code table (part)					
Character	**Decimal**	**Character**	**Decimal**	**Character**	**Decimal**
<Space>	32	I	73	R	82
A	65	J	74	S	83
B	66	K	75	T	84
C	67	L	76	U	85
D	68	M	77	V	86
E	69	N	78	W	87
F	70	O	79	X	88
G	71	P	80	Y	89
H	72	Q	81	Z	90

ACC	Location 450	OUTPUT

```
300  LDI   450
301  CMP    32
302  JPE   308
303  OUTCH
304  LDD   450
305  INC   ACC
306  STO   450
307  JMP   300
308  END

450       500

500        65
501        74
502        65
503        90
504        32
```

[5]