Computer Structure and Language

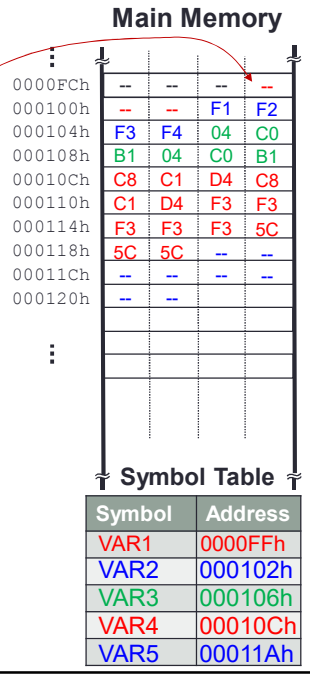
Hamid Sarbazi-Azad
Department of Computer Engineering
Sharif University of Technology (SUT)
Tehran, Iran

Character Processing

Any byte (00h … FFh) is known as character (not only EBCDIC code words of alphabets) in IBM machines.

We have several ways to define characters (bytes) in assembler.

```
        .....
* We define some byte variables.
* Lets assume Location Counter = 0000FFh, here.
VAR1    DS      3C
VAR2    DC      C'1234'
VAR3    DC      2X'4C0B1'
VAR4    DC      2C'HAM', 5X'F3',C'***'
VAR5    DS      8X
        .....
```



Symbol Table

Symbol	Address
VAR1	0000FFh
VAR2	000102h
VAR3	000106h
VAR4	00010Ch
VAR5	00011Ah

Character Processing (SS1 Instructions):

Move Character

Mnemonic: MVC S1(L),S2
MVC D1(L,B1),S2
MVC S1(L),D2(B2)
MVC D1(L,B1),D2(B2)

Operation: $M_{D1+(B1)} \leftarrow (M_{D2+(B2)})_{L\ bytes}$;

OPCODE: D2h

Note: All SS1 instructions (including MVC) work from left to right (i.e. lower address to higher address)

Character Processing (SS1 Instructions):

Move Character

Examples:

MVC VAR1(4),VAR3

Main Memory

0000FCh	--	--	--	--
000100h	--	--	F1	F2
000104h	F3	F4	04	C0
000108h	B1	04	C0	B1
00010Ch	C8	C1	D4	C8
000110h	C1	D4	F3	F3
000114h	F3	F3	F3	5C
000118h	5C	5C	--	--
00011Ch	--	--	--	--
000120h	--	--		

Symbol Table

Symbol	Address
VAR1	0000FFh
VAR2	000102h
VAR3	000106h
VAR4	00010Ch
VAR5	00011Ah

Character Processing (SS1 Instructions):

Move Character

Examples:

MVC VAR1(4),VAR3 after execution
MVC VAR1+2(3),VAR3-3

Main Memory

0000FCh	--	--	--	04
000100h	C0	B1	04	F2
000104h	F3	F4	04	C0
000108h	B1	04	C0	B1
00010Ch	C8	C1	D4	C8
000110h	C1	D4	F3	F3
000114h	F3	F3	F3	5C
000118h	5C	5C	--	--
00011Ch	--	--	--	--
000120h	--	--		

Symbol Table

Symbol	Address
VAR1	0000FFh
VAR2	000102h
VAR3	000106h
VAR4	00010Ch
VAR5	00011Ah

Character Processing (SS1 Instructions):

Move Character

Examples:

MVC VAR1(4),VAR3 after execution
MVC VAR1+2(3),VAR3-3 after execution
MVC VAR5(7),=C'ABCDEFG'

Main Memory

0000FCh	--	--	--	04
000100h	C0	F2	F3	F4
000104h	F3	F4	04	C0
000108h	B1	04	C0	B1
00010Ch	C8	C1	D4	C8
000110h	C1	D4	F3	F3
000114h	F3	F3	F3	5C
000118h	5C	5C	--	--
00011Ch	--	--	--	--
000120h	--	--		

Symbol Table

Symbol	Address
VAR1	0000FFh
VAR2	000102h
VAR3	000106h
VAR4	00010Ch
VAR5	00011Ah

Character Processing (SS1 Instructions):

Move Character

Examples:

```
MVC   VAR1(4),VAR3              after execution
MVC   VAR1+2(3),VAR3-3          after execution
MVC   VAR5(7),=C'ABCDEFG'       after execution
MVC   VAR3(3),=F'-1'
```

Main Memory

0000FCh	--	--	--	04
000100h	C0	F2	F3	F4
000104h	F3	F4	04	C0
000108h	B1	04	C0	B1
00010Ch	C8	C1	D4	C8
000110h	C1	D4	F3	F3
000114h	F3	F3	F3	5C
000118h	5C	5C	C1	C2
00011Ch	C3	C4	C5	C6
000120h	C7	--		

Symbol Table

Symbol	Address
VAR1	0000FFh
VAR2	000102h
VAR3	000106h
VAR4	00010Ch
VAR5	00011Ah

Character Processing (SS1 Instructions):

Move Character

Examples:

```
MVC   VAR1(4),VAR3              after execution
MVC   VAR1+2(3),VAR3-3          after execution
MVC   VAR5(7),=C'ABCDEF'        after execution
MVC   VAR3(3),=F'-1'            after execution
MVC   VAR1(1),=H'0'
```

Main Memory

0000FCh	--	--	--	04
000100h	C0	F2	F3	F4
000104h	F3	F4	FF	FF
000108h	FF	04	C0	B1
00010Ch	C8	C1	D4	C8
000110h	C1	D4	F3	F3
000114h	F3	F3	F3	5C
000118h	5C	5C	C1	C2
00011Ch	C3	C4	C5	C6
000120h	C7	--		

Symbol Table

Symbol	Address
VAR1	0000FFh
VAR2	000102h
VAR3	000106h
VAR4	00010Ch
VAR5	00011Ah

Character Processing (SS1 Instructions):

Move Character

Examples:

```
MVC   VAR1(4),VAR3          after execution
MVC   VAR1+2(3),VAR3-3      after execution
MVC   VAR5(7),=C'ABCDEF'    after execution
MVC   VAR3(3),=F'-1'        after execution
MVC   VAR1(1),=H'0'         after execution
MVC   VAR1+1(40),VAR1
```

Main Memory

0000FCh	--	--	--	00
000100h	C0	F2	F3	F4
000104h	F3	F4	FF	FF
000108h	FF	04	C0	B1
00010Ch	C8	C1	D4	C8
000110h	C1	D4	F3	F3
000114h	F3	F3	F3	5C
000118h	5C	5C	C1	C2
00011Ch	C3	C4	C5	C6
000120h	C7	--		

Symbol Table

Symbol	Address
VAR1	0000FFh
VAR2	000102h
VAR3	000106h
VAR4	00010Ch
VAR5	00011Ah

Character Processing (SS1 Instructions):

Move Character

Examples:

```
MVC   VAR1(4),VAR3          after execution
MVC   VAR1+2(3),VAR3-3      after execution
MVC   VAR5(7),=C'ABCDEF'    after execution
MVC   VAR3(3),=F'-1'        after execution
MVC   VAR1(1),=H'0'         after execution
MVC   VAR1+1(40),VAR1       after execution
```

Main Memory

0000FCh	--	--	--	00
000100h	00	00	00	00
000104h	00	00	00	00
000108h	00	00	00	00
00010Ch	00	00	00	00
000110h	00	00	00	00
000114h	00	00	00	00
000118h	00	00	00	00
00011Ch	00	00	00	00
000120h	00	00	00	00
	00	00	00	00

Symbol Table

Symbol	Address
VAR1	0000FFh
VAR2	000102h
VAR3	000106h
VAR4	00010Ch
VAR5	00011Ah

Character Processing (SS1 Instructions):

Move Character

Example 1:
Assembly instruction: MVC 10(9,9),1(7)
Operation: $M_{(R9)+10} \leftarrow (M_{(R7)+1})_{9\ bytes}$;
Machine code: D208900A7001

Example 2:
Assembly instruction: MVC VAR1+1(100),5(5) VAR1 = (R12)+10
Operation: $M_{(R12)+11} \leftarrow (M_{(R5)+5})_{100\ bytes}$;
Machine code: D263C00B5005

Example 3:
Assembly instruction: MVC VAR1-4(10),VAR2 VAR2 = (R12)+105
Operation: $M_{(R12)+6} \leftarrow (M_{(R12)+105})_{10\ bytes}$;
Machine code: D209C006C069

Character Processing (SS1 Instructions):

Compare Logical Character

Mnemonic: CLC S1(L),S2
CLC D1(L,B1),S2
CLC S1(L),D2(B2)
CLC D1(L,B1),D2(B2)

Operation: Compare $(M_{D1+(B1)})_{L\ bytes}$ with $(M_{D2+(B2)})_{L\ bytes}$ and update CC

OPCODE: D5h

Character Processing (SS1 Instructions):

Compare Logical Character

Example 1:
Assembly instruction: CLC 10(10,10),11(11)
Operation: Compare $(M_{(R10)+10})_{10\ bytes}$ and $(M_{(R11)+11})_{10\ bytes}$ and update CC;
Machine code: D509A00AB00B

Example 2:
Assembly instruction: CLC VAR1(20),2(3) VAR1 = (R12)+10
Operation: Compare $(M_{(R12)+10})_{20\ bytes}$ and $(M_{(R3)+2})_{20\ bytes}$ and update CC;
Machine code: D513C00A3002

Example 3:
Assembly instruction: CLC VAR1(2),VAR2 VAR2 = (R12)+101
Operation: Compare $(M_{(R12)+10})_{2\ bytes}$ and $(M_{(R12)+101})_{2\ bytes}$ and update CC;
Machine code: D501C00AC065

Character Processing (SS1 Instructions):

AND Character

Mnemonic: NC S1(L),S2
NC D1(L,B1),S2
NC S1(L),D2(B2)
NC D1(L,B1),D2(B2)

Operation: $M_{D1+(B1)} \leftarrow (M_{D1+(B1)})_{L\ bytes} \wedge (M_{D2+(B2)})_{L\ bytes}$ and update CC;

OPCODE: D4h

Note: All SS1 instructions (including NC) work from left to right (i.e. lower address to higher address)

Character Processing (SS1 Instructions):

AND Character

Example 1:
Assembly instruction: NC ␣(5,5),␣(5)
Operation: $M_{(R5)+5} \leftarrow (M_{(R5)+5})_{5\ bytes} \wedge (M_{(R5)+5})_{5\ bytes}$ and update CC;
Machine code: D4045005␣500␣

Example 2:
Assembly instruction: NC VAR1(20),2(3) VAR1 = (R12)+10
Operation: $M_{(R12)+10} \leftarrow (M_{(R12)+10})_{20\ bytes} \wedge (M_{(R3)+2})_{20\ bytes}$ and update CC;
Machine code: D413C00A3002

Example 3:
Assembly instruction: NC VAR1(2),VAR2 VAR2 = (R12)+101
Operation: $M_{(R12)+10} \leftarrow (M_{(R12)+10})_{2\ bytes} \wedge (M_{(R12)+101})_{2\ bytes}$ and update CC;
Machine code: D401C00AC065

Character Processing (SS1 Instructions):

OR Character

Mnemonic:
OC S1(L),S2
OC D1(L,B1),S2
OC S1(L),D2(B2)
OC D1(L,B1),D2(B2)

Operation: $M_{D1+(B1)} \leftarrow (M_{D1+(B1)})_{L\ bytes} \vee (M_{D2+(B2)})_{L\ bytes}$ and update CC;

OPCODE: D6h

Note: All SS1 instructions (including OC) work from left to right (i.e. lower address to higher address)

Character Processing (SS1 Instructions):

OR Character

Example 1:
Assembly instruction: OC 4(4,4),5(5)
Operation: $M_{(R4)+4} \leftarrow (M_{(R4)+4})_{4\ bytes} \vee (M_{(R5)+5})_{4\ bytes}$ and update CC;
Machine code: D60340045005

Example 2:
Assembly instruction: OC VAR1(2),20(3) VAR1 = (R12)+10
Operation: $M_{(R12)+10} \leftarrow (M_{(R12)+10})_{2\ bytes} \vee (M_{(R3)+20})_{2\ bytes}$ and update CC;
Machine code: D601C00A3014

Example 3:
Assembly instruction: OC VAR1-1(2),VAR2 VAR2 = (R12)+101
Operation: $M_{(R12)+9} \leftarrow (M_{(R12)+9})_{2\ bytes} \vee (M_{(R12)+101})_{2\ bytes}$ and update CC;
Machine code: D601C009C065

Character Processing (SS1 Instructions):

Exclusive-or Character

Mnemonic:
XC S1(L),S2
XC D1(L,B1),S2
XC S1(L),D2(B2)
XC D1(L,B1),D2(B2)

Operation: $M_{D1+(B1)} \leftarrow (M_{D1+(B1)})_{L\ bytes}$ xor $(M_{D2+(B2)})_{L\ bytes}$ and update CC;

OPCODE: D7h

Note: All SS1 instructions (including XC) work from left to right (i.e. lower address to higher address)

Character Processing (SS1 Instructions):

Exclusive-or Character

Example 1:
Assembly instruction: XC 4(4,4),5(5)
Operation: $M_{(R4)+4} \leftarrow (M_{(R4)+4})_{4\ bytes}$ **xor** $(M_{(R5)+5})_{4\ bytes}$ and update CC;
Machine code: D70340045005

Example 2:
Assembly instruction: XC VAR1(2),20(3) VAR1 = (R12)+10
Operation: $M_{(R12)+10} \leftarrow (M_{(R12)+10})_{2\ bytes}$ **xor** $(M_{(R3)+20})_{2\ bytes}$ and update CC;
Machine code: D701C00A3014

Example 3:
Assembly instruction: XC VAR1-3(3),VAR2+2 VAR2 = (R12)+101
Operation: $M_{(R12)+7} \leftarrow (M_{(R12)+7})_{3\ bytes}$ **xor** $(M_{(R12)+103})_{3\ bytes}$ and update CC;
Machine code: D702C007C067

Example 1: Write an assembly program to replace string "ALLAH" with "KHODA" in an input text of 1000 characters.

```
MYPROG  START  0
[Defining R12 as base register
 & initialize it to 6 → (R12) =6.]
        LA     2,TEXT
        LA     10,995(2)
LOOP    CLC    0(5,2),=C'ALLAH'
        BNE    OUT
        MVC    0(5,2),=C'KHODA'
        LA     2,4(2)
OUT     LA     2,1(2)
        CR     2,10
        BL     LOOP
[Returning to OS]
TEXT    DS     1000C
        END    MYPROG
```

Example 1: Write an assembly program to replace string "ALLAH" with "KHODA" in an input text of 1000 characters.

Address	Machine Code	Assembly Code		
000000		MYPROG	START	0
		[Defining R12 as base register & initialize it to 6 → (R12) =6.]		
000006	4120C02C		LA	2,TEXT
00000A	41A203E3		LA	10,995(2)
00000E	D5042000C414	LOOP	CLC	0(5,2),=C'ALLAH'
000014	4770C01C		BNE	OUT
000018	D2042000C419		MVC	0(5,2),=C'KHODA'
00001E	41220004		LA	2,4(2)
000022	41220001	OUT	LA	2,1(2)
000026	192A		CR	2,10
000028	4740C008		BL	LOOP
00002C		[Returning to OS]		
000032		TEXT	DS	1000C
			END	MYPROG
00041A	C1D3D3C1C8	=C'ALLAH'	DC	C'ALLAH'
00041F	D2C8D6C4C1	=C'KHODA'	DC	C'KHODA'
			END	MYPROG

Symbol Table

Symbol	B	Disp.
LOOP	C	008h
OUT	C	01Ch
TEXT	C	02Ch
=C'ALLAH'	C	414h
=C'KHODA'	C	419h

Character Processing (SI Instructions):

Move Immediate

Mnemonic: MVI S1,I2
MVI D1(B1),I2

Operation: $M_{D1+(B1)} \leftarrow I2;$

OPCODE: 92h

Example 1:

Assembly instruction: MVI VAR1,C'A' VAR1 = (R12)+200

Operation: $M_{(R12)+200} \leftarrow C1h;$

Machine code: 92C1C0C8

Character Processing (SI Instructions):

Move Immediate

Example 2:

Assembly instruction: MVI 11(11),11

Operation: $M_{(R11)+11} \leftarrow 0Bh$;

Machine code: 920BB00B

Example 3:

Assembly instruction: MVI A+5,B'10101' A = (R12)+200

Operation: $M_{(R12)+205} \leftarrow 15h$;

Machine code: 9215C0CD

Character Processing (SI Instructions):

Compare Logical Immediate

Mnemonic: CLI S1,I2
CLI D1(B1),I2

Operation: Compare ($M_{D1+(B1)}$) and I2 and update CC;

OPCODE: 95h

Example 1:

Assembly instruction: CLI VAR1,C'A' VAR1 = (R12)+200

Operation: Compare ($M_{(R12)+200}$) and C1h and update CC

Machine code: 95C1C0C8

Character Processing (SI Instructions):

Compare Logical Immediate

Example 2:

Assembly instruction: CLI 0(11),11

Operation: Compare ($M_{(R11)}$) and 0Bh and update CC;

Machine code: 950BB000

Example 3:

Assembly instruction: CLI A,B'10000101' A = (R12)+200

Operation: Compare ($M_{(R12)+200}$) and 85h and update CC;

Machine code: 9585C0C8

Character Processing (SI Instructions):

AND Immediate

Mnemonic: NI S1,I2
NI D1(B1),I2

Operation: $M_{D1+(B1)} \leftarrow (M_{D1+(B1)}) \wedge$ I2 and update CC;

OPCODE: 94h

Example 1:

Assembly instruction: NI VAR1,11 VAR1 = (R12)+200

Operation: $M_{(R12)+200} \leftarrow (M_{(R12)+200}) \wedge$ 0Bh and update CC;

Machine code: 940BC0C8

Character Processing (SI Instructions):

AND Immediate

Example 2:

Assembly instruction: NI 0(11),100

Operation: $M_{(R11)} \leftarrow (M_{(R11)}) \wedge 64h$ and update CC;

Machine code: 9464B000

Example 3:

Assembly instruction: NI A,B'111' A = (R12)+207

Operation: $M_{(R12)+207} \leftarrow (M_{(R12)+207}) \wedge 07h$ and update CC;

Machine code: 9407C0CF

Character Processing (SI Instructions):

OR Immediate

Mnemonic: OI S1,I2
OI D1(B1),I2

Operation: $M_{D1+(B1)} \leftarrow (M_{D1+(B1)}) \vee I2$ and update CC;

OPCODE: 96h

Example 1:

Assembly instruction: OI VAR1,255 VAR1 = (R12)+200

Operation: $M_{(R12)+200} \leftarrow (M_{(R12)+200}) \vee FFh$ and update CC;

Machine code: 96FFC0C8

Character Processing (SI Instructions):

OR Immediate

Example 2:

Assembly instruction: OI 0(11),100

Operation: $M_{(R11)} \leftarrow (M_{(R11)})$ v 64h and update CC;

Machine code: 9664B000

Example 3:

Assembly instruction: OI A,B'111' A = (R12)+207

Operation: $M_{(R12)+207} \leftarrow (M_{(R12)+207})$ v 07h and update CC;

Machine code: 9607C0CF

Character Processing (SI Instructions):

Exclusive-or Immediate

Mnemonic: XI S1,I2
XI D1(B1),I2

Operation: $M_{D1+(B1)} \leftarrow (M_{D1+(B1)})$ **xor** I2 and update CC;

OPCODE: 97h

Example 1:

Assembly instruction: XI VAR1,255 VAR1 = (R12)+200

Operation: $M_{(R12)+200} \leftarrow (M_{(R12)+200})$ **xor** FFh and update CC;

Machine code: 97FFC0C8

Character Processing (SI Instructions):

Exclusive-or Immediate

Example 2:

Assembly instruction: XI 0(11),100

Operation: $M_{(R11)} \leftarrow (M_{(R11)})$ **xor** 64h and update CC;

Machine code: 9764B000

Example 3:

Assembly instruction: XI A,B'111' A = (R12)+207

Operation: $M_{(R12)+207} \leftarrow (M_{(R12)+207})$ **xor** 07h and update CC;

Machine code: 9707C0CF

Example 2: Write an assembly program to concatenate two strings STR1 and STR
Note: En

Homework:
Translate this program into machine code.

```
STRING  START  0
        Defining R12 as base register
        & initialize it to 6  → (R12) =6.
        LA     2,STR1
        LA     3,STR3
LOP1    CLI    0(2),0
        BE     OUT1
        MVC    0(1,3),0(2)
        LA     2,1(2)
        LA     3,1(3)
        B      LOP1
OUT1    LA     2,STR2
LOP2    CLI    0(2),0
        BE     OUT2
        MVC    0(1,3),0(2)
        LA     2,1(2)
        LA     3,1(3)
        B      LOP2
OUT2    MVI    0(3),0
        Returning to OS
STR1    DC     C'xqisdsdqsidid',X'0'
STR2    DC     C'ahsaouaas',X'0'
STR3    DS     201C
        END    STRING
```

Character Processing (RX Instructions):

Insert Character

Mnemonic:
IC r1,S2(X2)
IC r1,S2
IC r1,D2(X2,B2)
IC r1,D2(X2)
IC r1,D2(,B2)

Operation: $r1_{7..0} \leftarrow (M_{D2+(B2)+(X2)})_{byte}$;

OPCODE: 43h

Note 1: Storage address S2 = D2 + (B2). On top of it, we have (X2) indexed to S2 to generate the final address.
Note 2: The least significant byte of r1 is loaded with a byte from memory.

Character Processing (RX Instructions):

Insert Character

Example 1:
Assembly instruction: IC 9,10(11,12)
Operation: $R9_{7..0} \leftarrow (M_{10+(R11)+(R12)})_{byte}$;
Machine code: 439BC00A

Example 2:
Assembly instruction: IC 2,ARRAY(B'111') ARRAY = (R12)+207
Operation: $R2_{7..0} \leftarrow (M_{207+(R7)+(R12)})_{byte}$;
Machine code: 4327C0CF

Example 3:
Assembly instruction: IC 4,0(3)
Operation: $R4_{7..0} \leftarrow (M_{(R3)})_{byte}$;
Machine code: 43430000

Character Processing (RX Instructions):

Store Character

Mnemonic:
STC r1,S2(X2)
STC r1,S2
STC r1,D2(X2,B2)
STC r1,D2(X2)
STC r1,D2(,B2)

Operation: $M_{D2+(B2)+(X2)} \leftarrow (r1)_{7..0}$;

OPCODE: 42h

Note 1: Storage address S2 = D2 + (B2). On top of it, we have (X2) indexed to S2 to generate the final address.
Note 2: The least significant byte of r1 is stored as a byte in memory.

Character Processing (RX Instructions):

Store Character

Example 1:
Assembly instruction: STC 9,10(11,12)
Operation: $M_{10+(R11)+(R12)} \leftarrow (R9)_{7..0}$;
Machine code: 429BC00A

Example 2:
Assembly instruction: STC 2,ARRAY(B'111') ARRAY = (R12)+207
Operation: $M_{207+(R7)+(R12)} \leftarrow (R2)_{7..0}$;
Machine code: 4227C0CF

Example 3:
Assembly instruction: STC 3,3(3)
Operation: $M_{(R3)+3} \leftarrow (R3)_{7..0}$;
Machine code: 42330003

Example 3: Write an assembly program to sort a100-element byte array ARR in ascending order (Gnome Sort).

```
SORTB  START  0                    .
[ Defining R12 as base register
& initialize it to 6  → (R12) =6. ]
       LA     3,99
       XR     2,2                  index
LOP    IC     4,ARR(2)
       IC     5,ARR+1(2)
       SLL    4,24
       SRA    4.24
       SLL    5,24
       SRA    5.24
       CR     4,5
       BC     B'1100',OUT   == BNH
       STC    4,ARR+1(2)
       STC    5,ARR(2)
       BCTR   2,0
       B      LOP
OUT    LA     2,1(2)
       CR     2,3
       BNE    LOP
[ Returning to OS ]
       DC     X'80'
ARR    DS     100C
       END    SORTB
```


Example 3: Write an assembly program to sort a100-element byte array ARR in ascending order (Gnome Sort).

Address	Machine Code	Assembly Code			
000000		SORTB	START	0	.
		[Defining R12 as base register & initialize it to 6]			
000006	41300063		LA	3,99	
00000A	1722		XR	2,2	index
00000C	4342C043	LOP	IC	4,ARR(2)	
000010	4352C044		IC	5,ARR+1(2)	
000014	89400018		SLL	4,24	
000018	8A400018		SRA	4.24	
00001C	89500018		SLL	5,24	
000020	8A500018		SRA	5.24	
000024	1945		CR	4,5	
000026	47C0C032		BC	B'1100',OUT	== BNH
00002A	4242C044		STC	4,ARR+1(2)	
00002E	4252C043		STC	5,ARR(2)	
000032	0620		BCTR	2,0	
000034	47F0C006		B	LOP	
000038	41220001	OUT	LA	2,1(2)	
00003C	1923		CR	2,3	
00003E	4770C006		BNE	LOP	
000042		[Returning to OS]			
000048	80		DC	X'80'	
000049		ARR	DS	100C	
			END	SORTB	

Symbol Table

Symbol	B	Disp.
LOP	C	006h
OUT	C	032h
ARR	C	043h

End of Slides