Midterm

Nathan Warner

Northern Illinois
University

Computer Science
Northern Illinois University
United States

Contents

Facts

- **Positive and negative hex numbers**: Leftmost hex digit 0-7: positive. 8-F: negative
- **Instructions and their lengths**: Look at the leftmost hex digit
  - **0,1,2,3**: 2 byte
  - **4-B**: 4 byte
  - **C-F**: 6 byte
- **Exceptions**
  - **S0C 1 (Operation)**: Invalid opcode / instruction
  - **SOC 4 (Protection)**: Accessed memory outside scope of program
  - **SOC 5 (Addressing)**: Invalid address
  - **SOC 6 (Specification)**: Operand not on FWB
- **Which instructions require FWB operands (might cause S0C6?)**: All RX
- **The byte in an SI instruction**: Given in one of four ways, a character, a two digit hex number, an eight bit binary number, or a decimal between 0 and 255, which will be converted to hex in the encoding

```
MVI   42(15),C'$'
MVI   42(15),X'5B'
MVI   42(15),B'01110110'
MVI   42(15),32
```

- **EQU**: EQU, or EQUates, assigns a value to a label

```
label    EQU    Expression
```

gives a label the value of the expression. Every occurrence of label will be treated as if it was the expression.

Equates are either typed above the CSECT or below the END

```
LOAD  EQU  L
```

Then, a load instruction can be written as

```
LOAD   3,NUM1
```

Before assembling the code, the Assembler replaces the label of the equates with the expression.

- **Carriage control**
  - **C' '**: Single space
  - **C'0'**: Double space
  - **C'-'**: Triple space
  - **C'1'**: Top of next page
- **Label that does not occupy storage**

```
label    DS    0H
```

- **Add one to register** R

```
LA   R,1(,R)
```

Decrement register R

```
BCTR R,0
```

Zero out register:

```
SR   R,R
```

FWB: Rightmost hex digit 0,4,8,C

HWB: Rightmost hex digit 0,2,4,6,8,A,C,E

DWB: Rightmost hex digit 0,8

LTORG starts on DWB

Max displacement: FFF, or 4095

Read loop:

```
         XREAD RECORD,80
LOOP1    BC    B'0111', ENDLOOP1
         XPRNT DETAIL,133
         XREAD RECORD,80
         BC    B'1111',LOOP1
ENDLOOP1
```

Instructions

2.1 RR

- **AR (1A)**: Sets condition code
- **SR (1B)**: Sets condition code
- **LR (18)**:
- **CR (19)**: Sets condition code
- **BCR (07)**:
- **MR (1C)**:
- **DR (1D)**
- **LTR (12) (Load and test register)**: Sets condition code
- **LCR (13) (Load complement register)**: Sets condition code

```
LCR   R1,R2
```

  If R2 contains n, R1 will contain $-n$

- **LNR (11) (Load negative register)**: Sets condition code

```
LNR   R1,R2
```

  Causes the negative of the absolute value of R2 to be loaded into R1

- **LPR (10) (Load positive register)**: Sets condition code

```
LPR   R1,R2
```

  Causes the absolute value of R2 to be loaded into R1. Overflow will occur if R2 contains the most negative number

- **BCTR (06) (Branch on count register)**: RR variant of BCT. Except, if R2 is R0, 1 is subtracted from R1, but no branch is taken

2.2 RX

- **A (5A)**: Sets condition code
- **S (5B)**: Sets condition code
- **L (58)**
- **ST (50)**
- **C (59)**: Sets condition code
- **BC (47)**

- **M (5C)**
- **D (5D)**
- **LA (41)**
- **BCT (07) (Branch on count)**:

```
BCT   R,D(X,B)
```

Causes contents of R to be decremented by one. If after this decrement R does not contain zero, a branch to the address D(X,B) is taken. If R does contain zero, the branch is not taken

2.3 SS (Storage to storage)

- **MVC (D2)**
- **CLC (D5)**: Sets condition code

2.4 SI (Storage to immediate byte)

- **MVI (92) (Move immediate)**: Replacement of the contents of the byte at D(B) with a copy of the immediate byte

```
MVI   D(B),byte
```

- **CLI (95) (Compare logical immediate)**: Sets condition code

Compares the byte at D(B) with the specified byte

2.5 X instructions

- **XDUMP**

```
XDUMP   D(X,B),Length, Any comments, notice the comma
XDUMP                Dump it all
```

- **XREAD**

```
XREAD   D(X,B),length
```

Note that the length is usually 80

- **XDECI**

```
o  XDECI   R,addr
```

- **XDECO**

```
o  XDECO   R,addr
```

  Note that XDECO requires character storage of length 12 bytes

- **XPRNT**

```
o  XPRNT   addr,len
```

  Note that len is usually 133

Condition codes

- **Numeric and character compares**
  - **Zero** if $a = b$
  - **One** if $a < b$
  - **Two** if $a > b$
- **LTR**
  - **Zero**: If the loaded value is zero
  - **One** If the loaded value is negative.
  - **Two**: If the loaded value is positive
- **LCR**
  - **Zero**: If the loaded value is zero
  - **One** If the loaded value is negative.
  - **Two**: If the loaded value is positive
  - **Three:** If there is overflow
- **LNR**
  - **Zero**: If the loaded value is zero
  - **One**: If the loaded value is negative (R2 was nonzero)
- **LPR**
  - **Zero**: If the loaded value is zero
  - **Two**: If the loaded value is positive
  - **Three**: If overflow occurred
- **XREAD**
  - **Zero**: Read success
  - **One**: No more to read
- **XDECI**
  - **Zero**: Number was converted to zero
  - **One:** Number was converted less than zero
  - **Two** Number was converted greater than zero
  - **Three**: Tried to convert invalid number

Encodings

- **RR**

$$OP_1OP_2R_1R_2$$

- **RX**

$$OP_1OP_2R_1I_1B_1D_1D_2D_3$$

- **SS**

$$OP_1OP_2L_1L_2B_1DDDB_2DDD$$

Note: L_1L_2 is the length minus 1

- **SI**

$$OP_1OP_2I_1I_2BDDD$$

- **BCR B'1111',14**: Encoded as

$$07FE$$

Extended mnemonics

Extended Mnemonic		Meaning	Replaces	
RX version	RR version		RX version	RR version
BH D(X,B)	BHR R	Branch on HIGH	BC B‘0010’,D(X,B)	BCR B‘0010’,R
BL D(X,B)	BLR R	Branch on LOW	BC B‘0100’,D(X,B)	BCR B‘0100’,R
BE D(X,B)	BER R	Branch on EQUAL	BC B‘1000’,D(X,B)	BCR B‘1000’,R
BNH D(X,B)	BNHR R	Branch on NOT HIGH	BC B‘1101’,D(X,B)	BCR B‘1101’,R
BNL D(X,B)	BNLR R	Branch on NOT LOW	BC B‘1011’,D(X,B)	BCR B‘1011’,R
BNE D(X,B)	BNER R	Branch on NOT EQUAL	BC B‘0111’,D(X,B)	BCR B‘0111’,R

Extended Mnemonic		Meaning	Replaces	
RX version	RR version		RX version	RR version
BP D(X,B)	BPR R	Branch on PLUS	BC B‘0010’,D(X,B)	BCR B‘0010’,R
BM D(X,B)	BMR R	Branch on MINUS	BC B‘0100’,D(X,B)	BCR B‘0100’,R
BZ D(X,B)	BZR R	Branch on ZERO	BC B‘1000’,D(X,B)	BCR B‘1000’,R
BO D(X,B)	BOR R	Branch on OVERFLOW	BC B‘0001’,D(X,B)	BCR B‘0001’,R
BNP D(X,B)	BNPR R	Branch on NOT PLUS	BC B‘1101’,D(X,B)	BCR B‘1101’,R
BNM D(X,B)	BNMR R	Branch on NOT MINUS	BC B‘1011’,D(X,B)	BCR B‘1011’,R
BNZ D(X,B)	BNZR R	Branch on NOT ZERO	BC B‘0111’,D(X,B)	BCR B‘0111’,R
BNO D(X,B)	BNOR R	Branch on NOT OVERFLOW	BC B‘1110’,D(X,B)	BCR B‘1110’,R