Computer Structure and Language

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

Decimal Numbers Representation

Decimal numbers are processed in IBM360 in two formats: **Zoned Decimal** (or Unpacked Decimal) or **Packed Decimal.**

In Zoned Decimal representation each byte contains one digit of the decimal number. Each digit has a left-side nibble of F.
For an n-digit number, we need n bytes. The last byte contains the least significant digit and a sign nibble D (Debit) for negative numbers and C (Credit) for positive numbers.
→decimal number $a_{n-1}a_{n-2}\ldots a_1a_0$ (or $+a_{n-1}a_{n-2}\ldots a_1a_0$) is shown as $Fa_{n-1}Fa_{n-2}\ldots Fa_1Ca_0$ and number $-a_{n-1}a_{n-2}\ldots a_1a_0$ is shown as $Fa_{n-1}Fa_{n-2}\ldots Fa_1Da_0$.

Example 1: Decimal number -37084 is shown in zoned-decimal representation as: F3F7F0F8D4 (5 bytes in memory).

Example 2: Decimal number 374 (or +374) is shown as: F3F7C4 (3 bytes in memory).

Decimal Numbers Representation

Decimal numbers are processed in IBM360 in two formats: **Zoned Decimal** (or Unpacked Decimal) or **Packed Decimal.**

In Packed Decimal representation each byte contains two digits of the decimal number. The last byte contains the least significant digit and the sign. For an n-digit number we need [(n+1)/2] bytes.
→decimal number $a_{n-1}a_{n-2}\ldots a_1a_0$ (or $+a_{n-1}a_{n-2}\ldots a_1a_0$) is shown as $a_{n-1}a_{n-2}\ \ldots\ a_2a_1\ a_0C$ and number $-a_{n-1}a_{n-2}\ldots a_1a_0$ is shown as $a_{n-1}a_{n-2}\ \ldots\ a_2a_1\ a_0D$.

Example 1: Decimal numbers -37084 is shown in packed decimal format as: 37084D (3 bytes).

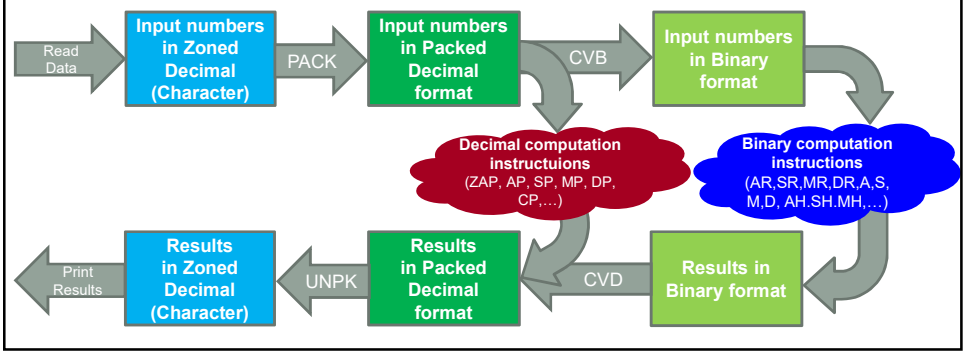
Example 2: Decimal numbers 1374 (or +1374) is shown in packed decimal format as: 01374C (3 bytes).

Decimal Numbers Processing

There are two ways to do decimal computation in IBM360:

Indirect: Convert decimal numbers into binary equivalents and do computation in binary format. The binary output must be converted to decimal to generate decimal output.

Direct: Directly do computation on decimal numbers and generate decimal output.

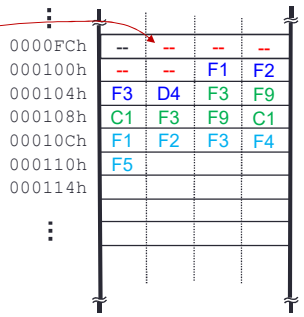


Decimal Numbers Definition

Zoned decimal numbers are defined as characters and hexadecimal numbers.

```
        .....
* We define some byte variables.
* Lets assume Location Counter = 0000FDh, here.
NUM1    DS      5X
NUM2    DC      X'F1F2F3D4'     -1234 Zoned decimal
NUM3    DC      2X'F3F9C1'      2 X (+391) Zoned dec.
NUM4    DC      C'12345'        +12345 Zoned decimal
        .....
```

Main Memory



Symbol Table

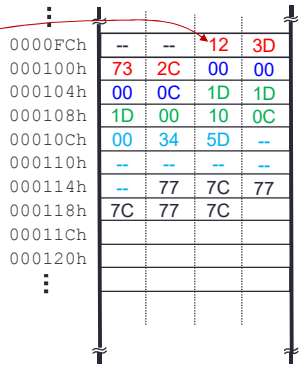
Symbol	Address
NUM1	0000FDh
NUM2	000102h
NUM3	000106h
NUM4	00010Ch

Decimal Numbers Definition

Packed decimal numbers are defined using data type P.

```
        .....
* We define some byte variables.
* Lets assume Location Counter = 0000FEh, here.
PN1     DC      P'-123', P'732'
NUM     DC      PL4'0'
HAM     DC      3P'-1',PL3'100'
P4      DC      XL3'345D'       = PL3'-345'
        DS      PL2, XL4
NUM2    DC      3P'+777'
        .....
```

Main Memory



Symbol Table

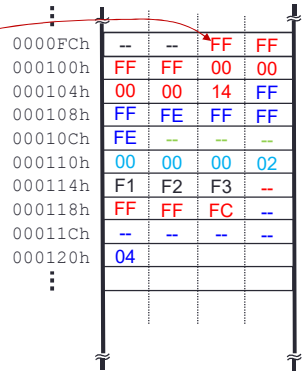
Symbol	Address
PN1	0000FEh
NUM	000102h
HAM	000106h
P4	00010Ch
NUM2	000115h

Length Attribute

Note that when we mention length, all alignment rules are disabled.

```
     .....
* We define some byte variables.
* Lets assume Location Counter = 0000FEh, here.
F1      DC      FL4'-1', FL3'0',FL2'20'
NUM     DC      2HL3'-2'
MAM     DS      0F
        DC      FL4'2'
NUM2    DC      CL3'1234'       Truncated from right
N       DS      0H
        DC      HL3'-4'
        DC      0D
        DC      FL1'260'        Truncated from left
        ....
```

Main Memory



Symbol Table

Symbol	Address
F1	0000FEh
NUM	000107h
MAM	000110h
NUM2	000114h
N	000117h

Note: Length attribute can be used as default for length parameter in SS1 and SS2 instructions when we do not mention data length.

Decimal Processing (SS2 Instructions):

Pack Zoned Decimal to Packed Decimal

Mnemonic: PACK S1(L1),S2(L2)
PACK D1(L1,B1),S2(L2)
PACK S1(L1),D2(L2,B2)
PACK D1(L1,B1),D2(L2,B2)

Operation: $M_{D1+(B1)}{}_{L1\ bytes} \leftarrow$ PACK $[(M_{D2+(B2)})_{L2\ bytes}]$;

OPCODE: F2h

$(M_{D2+(B2)}) =$ $Fa_{n-1}\ Fa_{n-2}\ \ldots\ Fa_2\ Fa_1\ Sa_0$ Sign nibble S = C or D

$M_{D1+(B1)} \leftarrow$ $a_{n-1}a_{n-2}\ldots a_2a_1\ a_0S$

Note: All SS2 instructions (including PACK) work from right to left (i.e. higher address to lower address)

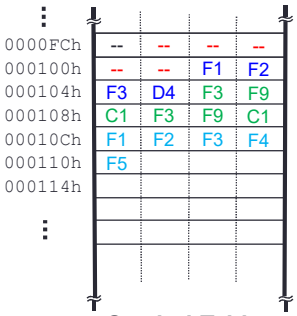
Decimal Processing (SS2 Instructions):

Pack Zoned Decimal to Packed Decimal

Examples:

PACK NUM1(5),NUM2(4)

Main Memory



Symbol Table

Symbol	Address
NUM1	0000FDh
NUM2	000102h
NUM3	000106h
NUM4	00010Ch

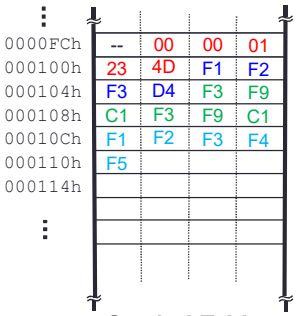
Decimal Processing (SS2 Instructions):

Pack Zoned Decimal to Packed Decimal

Examples:

PACK NUM1(5),NUM2(4) after execution
PACK NUM4(3),NUM4+3(2)

Main Memory



Symbol Table

Symbol	Address
NUM1	0000FDh
NUM2	000102h
NUM3	000106h
NUM4	00010Ch

Decimal Processing (SS2 Instructions):

Pack Zoned Decimal to Packed Decimal

Examples:

PACK NUM1(5),NUM2(4) after execution
PACK NUM4(3),NUM4+3(2) after execution
PACK NUM3(3),NUM3+3(3)

Main Memory

0000FCh	--	00	00	01
000100h	23	4D	F1	F2
000104h	F3	D4	F3	F9
000108h	C1	F3	F9	C1
00010Ch	00	04	5F	F4
000110h	F5			
000114h				

Symbol Table

Symbol	Address
NUM1	0000FDh
NUM2	000102h
NUM3	000106h
NUM4	00010Ch

Decimal Processing (SS2 Instructions):

Pack Zoned Decimal to Packed Decimal

Examples:

PACK NUM1(5),NUM2(4) after execution
PACK NUM4(3),NUM4+3(2) after execution
PACK NUM3(3),NUM3+3(3) after execution
PACK NUM1(2),NUM2(4)

Main Memory

0000FCh	--	00	00	01
000100h	23	4D	F1	F2
000104h	F3	D4	00	39
000108h	1C	F3	F9	C1
00010Ch	00	04	5F	F4
000110h	F5			
000114h				

Symbol Table

Symbol	Address
NUM1	0000FDh
NUM2	000102h
NUM3	000106h
NUM4	00010Ch

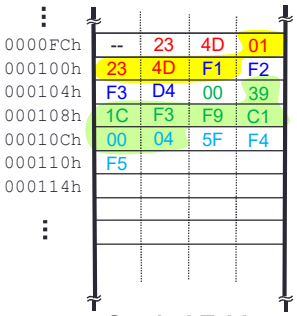
Decimal Processing (SS2 Instructions):

Pack Zoned Decimal to Packed Decimal

Examples:

PACK NUM1(5),NUM2(4) after execution
PACK NUM4(3),NUM4+3(2) after execution
PACK NUM3(3),NUM3+3(3) after execution
PACK NUM1(2),NUM2(4) after execution
PACK NUM2+5(7),NUM1+2(4)

Main Memory



Symbol Table

Symbol	Address
NUM1	0000FDh
NUM2	000102h
NUM3	000106h
NUM4	00010Ch

Decimal Processing (SS2 Instructions):

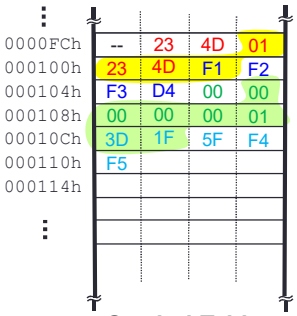
Pack Zoned Decimal to Packed Decimal

Examples:

PACK NUM1(5),NUM2(4) after execution
PACK NUM4(3),NUM4+3(2) after execution
PACK NUM3(3),NUM3+3(3) after execution
PACK NUM1(2),NUM2(4) after execution
PACK NUM2+5(7),NUM1+2(4) after execution

Note 1: If S1 does not have enough space for the packed number, digits are truncated from left.
Note 2: There might be overlaps between S1 and S2. So you must be careful and do the packing from right to left.

Main Memory



Symbol Table

Symbol	Address
NUM1	0000FDh
NUM2	000102h
NUM3	000106h
NUM4	00010Ch

Decimal Processing (SS2 Instructions):

Pack Zoned Decimal to Packed Decimal

Example 1:
Assembly instruction: PACK 10(10,10),11(11,11)
Operation: Pack $(M_{(R11)+11})_{11\ bytes}$ and store it into $M_{(R10)+10}$ (10 bytes);
Machine code: F29AA00AB00B

Example 2:
Assembly instruction: PACK VAR1(8),2(3,5) VAR1 = (R12)+100
Operation: Pack $(M_{(R5)+2})_{3\ bytes}$ and store it into $M_{(R12)+100}$ (8 bytes);
Machine code: F272C0645002

Example 3:
Assembly instruction: PACK VAR1(2),VAR2(4) VAR2 = (R12)+101
Operation: Pack $(M_{(R12)+101})_{4\ bytes}$ and store it into $M_{(R12)+100}$ (2 bytes);
Machine code: F213C064C065

Decimal Processing (SS2 Instructions):

Unpack Packed Decimal to Zoned Decimal

Mnemonic: UNPK S1(L1),S2(L2)
UNPK D1(L1,B1),S2(L2)
UNPK S1(L1),D2(L2,B2)
UNPK D1(L1,B1),D2(L2,B2)
Operation: $M_{D1+(B1)_{L1\ bytes}} \leftarrow$ Unpack $[(M_{D2+(B2)})_{L2\ bytes}]$;
OPCODE: F3h

$(M_{D2+(B2)})$ = $a_{n-1}a_{n-2} \ldots a_2a_1a_0S$ Sign nibble S = C or D

$M_{D1+(B1)} \leftarrow$ $Fa_{n-1}\ Fa_{n-2}\ \ldots\ Fa_2\ Fa_1\ Sa_0$

Note: All SS2 instructions (including UNPK) work from right to left (i.e. higher address to lower address)

Decimal Processing (SS2 Instructions):

Unpack Packed Decimal to Zoned Decimal

Examples:

UNPK PN1-2(6),NUM(4)

Main Memory

0000FCh	--	--	12	3D
000100h	73	2C	00	00
000104h	00	0C	1D	1D
000108h	1D	00	10	0C
00010Ch	00	34	5D	--
000110h	--	--	--	--
000114h	--	77	7C	77
000118h	7C	77	7C	
00011Ch				
000120h				

Symbol Table

Symbol	Address
PN1	0000FEh
NUM	000102h
HAM	000106h
P4	00010Ch
NUM2	000115h

Decimal Processing (SS2 Instructions):

Unpack Packed Decimal to Zoned Decimal

Examples:

UNPK PN1-2(6),NUM(4)
UNPK NUM2-5(5),P4+1(2)

after execution

Main Memory

0000FCh	F0	F0	F0	F0
000100h	F0	C0	00	00
000104h	00	0C	1D	1D
000108h	1D	00	10	0C
00010Ch	00	34	5D	--
000110h	--	--	--	--
000114h	--	77	7C	77
000118h	7C	77	7C	
00011Ch				
000120h				

Symbol Table

Symbol	Address
PN1	0000FEh
NUM	000102h
HAM	000106h
P4	00010Ch
NUM2	000115h

Decimal Processing (SS2 Instructions):

Unpack Packed Decimal to Zoned Decimal

Examples:

UNPK PN1-2(6),NUM(4) after execution
UNPK NUM2-5(5),P4+1(2) after execution
UNPK HAM(2),HAM+3(3)

Main Memory

0000FCh	F0	F0	F0	F0
000100h	F0	C0	00	00
000104h	00	0C	1D	1D
000108h	1D	00	10	0C
00010Ch	00	34	5D	--
000110h	F0	F0	F3	F4
000114h	D5	77	7C	77
000118h	7C	77	7C	
00011Ch				
000120h				

Symbol Table

Symbol	Address
PN1	0000FEh
NUM	000102h
HAM	000106h
P4	00010Ch
NUM2	000115h

Decimal Processing (SS2 Instructions):

Unpack Packed Decimal to Zoned Decimal

Examples:

UNPK PN1-2(6),NUM(4) after execution
UNPK NUM2-5(5),P4+1(2) after execution
UNPK HAM(2),HAM+3(3) after execution
UNPK HAM(7),HAM+2(5)

Main Memory

0000FCh	F0	F0	F0	F0
000100h	F0	C0	00	00
000104h	00	0C	F0	C0
000108h	1D	00	10	0C
00010Ch	00	34	5D	--
000110h	F0	F0	F3	F4
000114h	D5	77	7C	77
000118h	7C	77	7C	
00011Ch				
000120h				

Symbol Table

Symbol	Address
PN1	0000FEh
NUM	000102h
HAM	000106h
P4	00010Ch
NUM2	000115h

Decimal Processing (SS2 Instructions):

Unpack Packed Decimal to Zoned Decimal

Examples:

UNPK PN1-2(6),NUM(4) after execution
UNPK NUM2-5(5),P4+1(2) after execution
UNPK HAM(2),HAM+3(3) after execution
UNPK HAM(7),HAM+2(5) after execution
UNPK NUM2(۶),NUM2(۳)

Main Memory

0000FCh	F0	F0	F0	F0
000100h	F0	C0	00	00
000104h	00	0C	FF	F0
000108h	FF	F0	F0	FC
00010Ch	00	F4	D5	--
000110h	F0	F0	F3	F4
000114h	D5	F7	F7	C7
000118h	7C	77	7C	
00011Ch				
000120h				

Symbol Table

Symbol	Address
PN1	0000FEh
NUM	000102h
HAM	000106h
P4	00010Ch
NUM2	000115h

Decimal Processing (SS2 Instructions):

Unpack Packed Decimal to Zoned Decimal

Examples:

UNPK PN1-2(6),NUM(4) after execution
UNPK NUM2-5(5),P4+1(2) after execution
UNPK HAM(2),HAM+3(3) after execution
UNPK HAM(7),HAM+2(5) after execution
UNPK NUM2(۶),NUM2(۳) after execution

Main Memory

0000FCh	F0	F0	F0	F0
000100h	F0	C0	00	00
000104h	00	0C	FF	F0
000108h	FF	F0	F0	FC
00010Ch	00	F4	D5	--
000110h	F0	F0	F3	F4
000114h	D5	F7	F7	FC
000118h	F7	F7	7C	
00011Ch				
000120h				

Symbol Table

Symbol	Address
PN1	0000FEh
NUM	000102h
HAM	000106h
P4	00010Ch
NUM2	000115h

Decimal Processing (SS2 Instructions):

Unpack Packed Decimal to Zoned Decimal

Example 1:
Assembly instruction: UNPK 1(2,3),4(5,6)
Operation: Unpack $(M_{(R6)+4})_{5\ bytes}$ and store it into $M_{(R3)+1}$ (2 bytes);
Machine code: F31430016004

Example 2:
Assembly instruction: UNPK VAR1(8),5(5,5) VAR1 = (R12)+100
Operation: Unpack $(M_{(R5)+5})_{5\ bytes}$ and store it into $M_{(R12)+100}$ (8 bytes);
Machine code: F374C0645005

Example 3:
Assembly instruction: UNPK VAR1(2),VAR2(4) VAR2 = (R12)+101
Operation: Unpack $(M_{(R12)+101})_{4\ bytes}$ and store it into $M_{(R12)+100}$ (2 bytes);
Machine code: F313C064C065

Decimal Processing (SS2 Instructions):

Add Packed Decimal

Mnemonic: AP S1(L1),S2(L2)
AP D1(L1,B1),S2(L2)
AP S1(L1),D2(L2,B2)
AP D1(L1,B1),D2(L2,B2)

Operation: $M_{D1+(B1)_{L1\ bytes}} \leftarrow (M_{D1+(B1)})_{L1\ bytes} + (M_{D2+(B2)})_{L2\ bytes}$ and update CC;

OPCODE: FAh

Note: All SS2 instructions (including AP) work from right to left (i.e. higher address to lower address)

Decimal Processing (SS2 Instructions):

Add Packed Decimal

Examples:

AP PN1(2),PN1+2(2)

Main Memory

0000FCh	--	--	12	3D
000100h	73	2C	00	00
000104h	00	0C	1D	1D
000108h	1D	00	10	0C
00010Ch	00	34	5D	--
000110h	--	--	--	--
000114h	--	77	7C	77
000118h	7C	77	7C	
00011Ch				
000120h				

Symbol Table

Symbol	Address
PN1	0000FEh
NUM	000102h
HAM	000106h
P4	00010Ch
NUM2	000115h

Decimal Processing (SS2 Instructions):

Add Packed Decimal

Examples:

AP PN1(2),PN1+2(2)
AP NUM(4),NUM2+4(2)

after execution

Main Memory

0000FCh	--	--	60	9C
000100h	73	2C	00	00
000104h	00	0C	1D	1D
000108h	1D	00	10	0C
00010Ch	00	34	5D	--
000110h	--	--	--	--
000114h	--	77	7C	77
000118h	7C	77	7C	
00011Ch				
000120h				

Symbol Table

Symbol	Address
PN1	0000FEh
NUM	000102h
HAM	000106h
P4	00010Ch
NUM2	000115h

Decimal Processing (SS2 Instructions):

Add Packed Decimal

Examples:

AP PN1(2),PN1+2(2) after execution
AP NUM(4),NUM2+4(2) after execution
AP HAM(1),P4(3)

Main Memory

0000FCh	--	--	60	9C
000100h	73	2C	00	00
000104h	77	7C	1D	1D
000108h	1D	00	10	0C
00010Ch	00	34	5D	--
000110h	--	--	--	--
000114h	--	77	7C	77
000118h	7C	77	7C	
00011Ch				
000120h				

Symbol Table

Symbol	Address
PN1	0000FEh
NUM	000102h
HAM	000106h
P4	00010Ch
NUM2	000115h

Decimal Processing (SS2 Instructions):

Add Packed Decimal

Examples:

AP PN1(2),PN1+2(2) after execution
AP NUM(4),NUM2+4(2) after execution
AP HAM(1),P4(3) after execution

Assuming base register is R12 and (R12) = 16, the machine code of above instructions are:

AP PN1(2),PN1+2(2) FA11C0EEC0F0
AP NUM(4),NUM2+4(2) FA31C0F2C109
AP HAM(1),P4(3) FA02C0F6C0FC

Main Memory

0000FCh	--	--	60	9C
000100h	73	2C	00	00
000104h	77	7C	6D	1D
000108h	1D	00	10	0C
00010Ch	00	34	5D	--
000110h	--	--	--	--
000114h	--	77	7C	77
000118h	7C	77	7C	
00011Ch				
000120h				

Symbol Table

Symbol	Address
PN1	0000FEh
NUM	000102h
HAM	000106h
P4	00010Ch
NUM2	000115h

Decimal Processing (SS2 Instructions):

Add Packed Decimal

Example 1:
Assembly instruction: AP 2(2,2),3(3,3)
Operation: $M_{(R2)+2}$ ← $(M_{(R2)+2})_{2\ bytes}$ + $(M_{(R3)+3})_{3\ bytes}$ and update CC;
($M_{(R2)+2}$: 2 bytes)

Machine code: FA1220023003

Example 2:
Assembly instruction: AP VAR1(8),0(5,5) VAR1 = (R12)+100
Operation: $M_{(R12)+100}$ ← $(M_{(R12)+100})_{8\ bytes}$ + $(M_{(R5)})_{5\ bytes}$ and update CC;
($M_{(R12)+100}$: 8 bytes)

Machine code: FA74C0645000

Decimal Processing (SS2 Instructions):

Subtract Packed Decimal

Mnemonic:
SP S1(L1),S2(L2)
SP D1(L1,B1),S2(L2)
SP S1(L1),D2(L2,B2)
SP D1(L1,B1),D2(L2,B2)

Operation: $M_{D1+(B1)}$ ← $(M_{D1+(B1)})_{L1\ bytes}$ - $(M_{D2+(B2)})_{L2\ bytes}$ and update CC;
($M_{D1+(B1)}$: L1 bytes)

OPCODE: FBh

Note: All SS2 instructions (including SP) work from right to left (i.e. higher address to lower address)

Decimal Processing (SS2 Instructions):

Subtract Packed Decimal

Example 1:
Assembly instruction: SP 0(2,2),0(3,3)
Operation: $M_{(R2)}$ ← $(M_{(R2)})_{2\ bytes}$ - $(M_{(R3)})_{3\ bytes}$ and update CC;
($M_{(R2)}$: 2 bytes)
Machine code: FB1220003000

Example 2:
Assembly instruction: SP VAR1(8),0(14,5) VAR1 = (R12)+100
Operation: $M_{(R12)+100}$ ← $(M_{(R12)+100})_{8\ bytes}$ - $(M_{(R5)})_{14\ bytes}$ and update CC;
($M_{(R12)+100}$: 8 bytes)
Machine code: FB7DC0645000

Example 3:
Assembly instruction: SP VAR1-2(2),VAR2+2(5) VAR2 = (R12)+10
Operation: $M_{(R12)+98}$ ← $(M_{(R12)+98})_{2\ bytes}$ - $(M_{(R12)+12})_{5\ bytes}$ and update CC;
($M_{(R12)+98}$: 2 bytes)
Machine code: FB14C062C00C

Decimal Processing (SS2 Instructions):

Compare Packed Decimal

Mnemonic:
CP S1(L1),S2(L2)
CP D1(L1,B1),S2(L2)
CP S1(L1),D2(L2,B2)
CP D1(L1,B1),D2(L2,B2)

Operation: Realize $(M_{D1+(B1)})_{L1\ bytes}$ - $(M_{D2+(B2)})_{L2\ bytes}$ and update CC;

OPCODE: F9h

Note: All SS2 instructions (including CP) work from right to left (i.e. higher address to lower address)

Decimal Processing (SS2 Instructions):

Compare Packed Decimal

Example 1:
Assembly instruction: CP 1(1,1),4(4,4)
Operation: Realize $(M_{(R1)+1})_{1\ bytes}$ - $(M_{(R4)+4})_{4\ bytes}$ and update CC;
Machine code: F90310014004

Example 2:
Assembly instruction: CP V1(8),0(1,5) V1 = (R12)+20
Operation: Realize $(M_{(R12)+20})_{8\ bytes}$ - $(M_{(R5)})_{1\ byte}$ and update CC;
Machine code: F970C0145000

Example 3:
Assembly instruction: CP V1-5(2),VAR2(11) VAR2 = (R12)+10
Operation: Realize $(M_{(R12)+15})_{2\ bytes}$ - $(M_{(R12)+10})_{11\ bytes}$ and update CC;
Machine code: F91AC00FC00A

Decimal Processing (SS2 Instructions):

Zero and Add Packed Decimal

Mnemonic:
ZAP S1(L1),S2(L2)
ZAP D1(L1,B1),S2(L2)
ZAP S1(L1),D2(L2,B2)
ZAP D1(L1,B1),D2(L2,B2)

Operation:
M_{S1} ($_{L1\ bytes}$) $\leftarrow$ 0;
M_{S1} ($_{L1\ bytes}$) $\leftarrow$ $(M_{S1})_{L1\ bytes}$ + $(M_{S2})_{L2\ bytes}$ and update CC;

OPCODE: F8h

Note: All SS2 instructions (including ZAP) work from right to left (i.e. higher address to lower address)

Decimal Processing (SS2 Instructions):

Zero and Add Packed Decimal

Example 1:
Assembly instruction: ZAP 1(1,1),0(2,4) Suppose $(M_{(R4)}) = 012C$
Operation: $M_{(R1)+1} \leftarrow 2C$; CC ←11;
Machine code: F80110014000

Example 2:
Assembly instruction: ZAP V1(2),=P'-1' V1 = (R12)+20
Operation: $M_{(R12)+20} \leftarrow 0 \cdot 1D$; CC ←01; =P'-1' = (R12)+200
Machine code: F810C014C0C8

Decimal Processing (SS2 Instructions):

Multiply Packed Decimal

Mnemonic: MP S1(L1),S2(L2)
MP D1(L1,B1),S2(L2)
MP S1(L1),D2(L2,B2)
MP D1(L1,B1),D2(L2,B2)

Operation: $M_{S1}{}_{L1\ bytes} \leftarrow (M_{S1})_{L1\ bytes} \times (M_{S2})_{L2\ bytes}$ and update CC;

OPCODE: FCh

Note: All SS2 instructions (including MP) work from right to left (i.e. higher address to lower address)

Decimal Processing (SS2 Instructions):

Multiply Packed Decimal

Example 1:
Assembly instruction: MP 1(3,1),0(2,4)

$(M_{(R4)}) = 012C$
$(M_{(R1)+1}) = 00010D$

Operation: $M_{(R1)+1} \leftarrow 00120D$; CC ←01;

Machine code: FC2110014000

Example 2:
Assembly instruction: MP V1+2(2),=P'-8'

V1 = (R12)+20,
$(M_{V1}) = 0000012D$
=P'-8' = (R12)+30

Operation: $M_{(R12)+20} \leftarrow 0000$**096C**; CC ←10;

Machine code: FC10C016C01E

Decimal Processing (SS2 Instructions):

Divide Packed Decimal

Mnemonic:
DP S1(L1),S2(L2)
DP D1(L1,B1),S2(L2)
DP S1(L1),D2(L2,B2)
DP D1(L1,B1),D2(L2,B2)

Operation: $M_{S1}{}_{L1\text{-}L2\ bytes} \leftarrow (M_{S1})_{L1\ bytes} / (M_{S2})_{L2\ bytes}$; and update CC;

$M_{S1+L1\text{-}L2}{}_{L2\ bytes} \leftarrow$ Remainder;

OPCODE: FDh

Note: All SS2 instructions (including DP) work from right to left (i.e. higher address to lower address)

Decimal Processing (SS2 Instructions):

Divide Packed Decimal

Example 1:
Assembly instruction: DP 0(3,1),0(2,4)

$(M_{(R1)}) = 00120C$,
$(M_{(R4)}) = 020D$

Operation: $M_{(R1)} \leftarrow 6D000C$; CC ←01;

Machine code: FD2110004000

Example 2:
Assembly instruction: DP V1(4),=P'5'

V1 = (R12)+10,
$(M_{V1}) = 0001004C$
=P'5' = (R12)+20

Operation: $M_{(R12)+10} \leftarrow 00200C4C$; CC ←10;

Machine code: FD30C00AC014

Decimal Processing (RX Instructions):

Convert Decimal to Binary

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

Operation: r1 ← Binary $[(M_{D2+(B2)+(X2)})_{8\ bytes}]$;

OPCODE: 4Fh

Note: Storage address S2 = D2 + (B2). On top of it, we have (X2) indexed to S2 to generate the final address. The final address must be a double-word address (8X).

Decimal Processing (RX Instructions):

Convert Decimal to Binary

Example 1:
Assembly instruction: CVB 3,NUMDEC

NUMDEC is a double-word with address (R12)+32, and (M_{NUMDEC}) = 00..0200C

Operation: R3 ← 000000C8h;

Machine code: 4F30C020

Example 2:
Assembly instruction: CVB 2,3(4,5)

Double-word stored in $M_{(R4)+(R5)+3}$ contains 00..0100D

Operation: R2 ← FFFFFF9Ch;

Machine code: 4F245003

Decimal Processing (RX Instructions):

Convert Binary to Decimal

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

Operation: $M_{D2+(B2)+(X2)}$ ← Decimal [(r1)] $_{8\ bytes}$;

OPCODE: 4Eh

Note: Storage address S2 = D2 + (B2). On top of it, we have (X2) indexed to S2 to generate the final address. The final address must be a double-word address (8X).

Decimal Processing (RX Instructions):

Convert Binary to Decimal

Example 1:
Assembly instruction: CVD 3,NUMDEC NUMDEC is a double-word with address (R12)+12 and (R3) = FFFFFF90h

Operation: $M_{(R12)+12} \leftarrow 000000000000070D;$

Machine code: 4E30C00C

Example 2:
Assembly instruction: CVD 7,0(10,11) Location $M_{(R10)+(R11)}$ is a double-word and (R7) = 00000101h.

Operation: $M_{(R10)+(R11)} \leftarrow 000000000000257C;$

Machine code: 4E7AB000

Example 1: Write an assembly program to compute convolution of vector A of 100 packed decimal elements (each 3 bytes) and vector B of 100 half-words. The result must be stored in the 7-byte packed decimal CONV.

```
PROG   START  0
[Defining R12 as base register
& initialize it to 6 → (R12) =6.]
       XR     2,2      index on B
       LA     3,A      pointer to A
       LA     4,100    counter
LOOP   LH     5,B(2)
       CVD    5,TEMP
       MP     TEMP(8),0(3,3)
       AP     CONV(7),TEMP(8)
       LA     2,2(2)
       LA     3,3(3)
       BCT    4,LOOP
[Returning to OS]
TEMP   DS     D
CONV   DC     PL7'0'
A      DC     PL3'56', PL3'-2', ….
B      DS     100H
       END    PROG
```

Example 1: Write an assembly program to compute convolution of vector A of 100 packed decimal elements (each 3 bytes) and vector B of 100 half-words. The result must be stored in the 7-byte packed decimal CONV.

Address	Machine Code		Assembly Code		
000000		PROG	START	0	
			Defining R12 as base register & initialize it to 6 → (R12) =6.		
000006	1722		XR	2,2	index on B
000008	4130C041		LA	3,A	pointer to A
00000C	41400064		LA	4,100	counter
000010	4852C16E	LOOP	LH	5,B(2)	
000014	4E50C032		CVD	5,TEMP	
000018	FC72C0323000		MP	TEMP(8),0(3,3)	
00001E	FA67C03AC032		AP	CONV(7),TEMP(8)	
000024	41220002		LA	2,2(2)	
000028	41330003		LA	3,3(3)	
00002C	4640C00A		BCT	4,LOOP	
000030			Returning to OS	6 bytes	
000038		TEMP	DS	D	
000040	0000000000000C	CONV	DC	PL7'0'	
000047	00056C 00002D	A	DC	PL3'56', PL3'-2', ….	
000174		B	DS	100H	
			END	PROG	

Symbol Table

Symbol	B	Disp.
LOOP	C	00Ah
TEMP	C	032h
CONV	C	03Ah
A	C	041h
B	C	16Eh

Example 2: Write an assembly program to extract numbers in a character string (ended by null) LINE, add them together, and store the result in character string RESULT.

```
SUMCHAR  START  0
| Defining R12 as base register |
| & initialize it to 6 → (R12) =6. |
```

```
| Returning to OS |
TEMP     DS  PL4
SUM      DC  PL6'0'
LINE     DC  C'-123  22  +401  -11  …  220',X'0'
SIGN     DS  C
RESULT   DS  12C,X'0'
         END     SUMCHAR
```

Example 2: Write an assembly program to extract numbers in a character string (ended by null) LINE, add them together, and store the result in character string RESULT.

```
SUMCHAR  START  0
| Defining R12 as base register  |
| & initialize it to 6  → (R12) =6.  |
         LA      2,LINE      pointer
LOOP     CLI     0(2),C' '
         BNE     NUL1
         LA      2,1(2)
         B       LOOP
NUL1     CLI     0(2),0
         BE      ENDP
         ZAP     SIGN,=P'1'
         CLI     0(2),C'-'
         BNE     POS
         ZAP     SIGN(1),=P'-1'
         LA      2,1(2)
         B       DIGIT1
POS      CLI     0(2),C'+'
         BNE     DIGIT1
         LA      2,1(2)
DIGIT1   LR      3,2
MORE     CLI     0(2),C' '
         BE      DIGITL
         CLI     0(2),0
         BE      DIGITL
         LA      2,1(2)
         B       MORE
DIGITL   LR      4,2
         SR      4,3
         LA      4,47(4)     R4 ← (4-1)*16+(R4)-1
         STC     4,PAK+1
PAK      PACK    TEMP(4),0(1,3)
         MP      TEMP(4),SIGN(1)
         AP      SUM(6),TEMP(4)
         B       LOOP
ENDP     XR      2,2
         UNPK    RESULT+1(11),SUM(6)
         CP      SUM(6),=P'-1'
         BH      PLUS
         MVI     RESULT,C'-'
         LA      2,1
PLUS     OI      RESULT+11,X'F0'
         LA      2,RESULT(2)
LAST     CLI     0(2),X'F0'     ==C'0'
         BNE     FINISH
         MVC     0(12,2),1(2)
         B       LAST
FINISH   CLI     0(2),0
         BNE     FIN
         MVI     0(2),C'0'
FIN
| Returning to OS |
TEMP     DS  PL4
SUM      DC  PL6'0'
LINE     DC  C'-123  22  +401  -11  …  220',X'0'
SIGN     DS  C
RESULT   DS  12C,X'0'
         END     SUMCHAR
```


Example 2: Write an assembly program to extract numbers in a character string (ended by null) LINE, add them together, and store the result in character string RESULT.

Homework:
Translate this program into machine code.

End of Slides