SM7831

Decimal Operation Processor

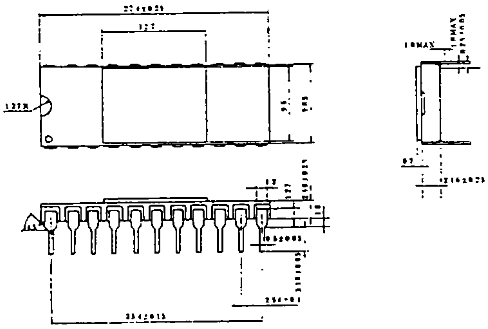
■ OVERVIEW

The SM7831 is a CMOS-LSI arithmetic processor developed for decimal long-word arithmetic. This LSI functions as the arithmetic co-processor and dramatically improves the cost performance of a microcomputer. The SM7831 fabricated in the CMOS process requires only an extremely low power to execute various arithmetic operations..

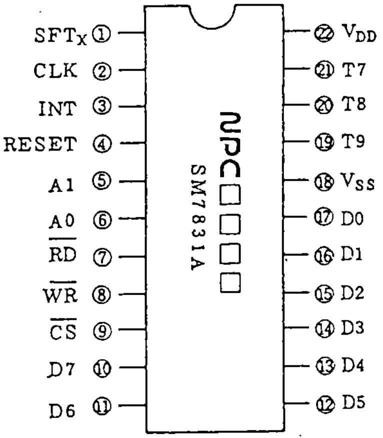
■ FEATURES

- High-speed execution of decimal arithmetic operations, such as addition, subtraction, multiplication, division and square root extraction
- Easy connection with the CPU
  Data transfer ... program transfer
  End of operation ... interrupt and status read
- The number of arithmetic digits can be selected by the program from 16 digits, 14 digits and 12 digits.
- Convenient register control commands
  Clear, Move, Exchange, Zero Sense, Normalize, Shift Right, Shift Left
- Low power consumption 10 mA Typ.
- CMOS single 5 V power supply
- 22-pin ceramic DIP

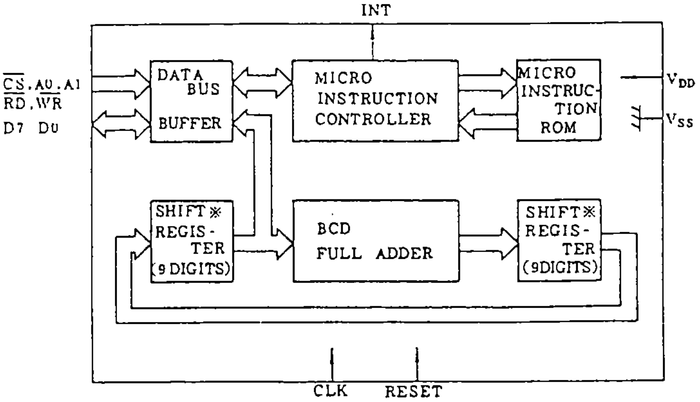
■ OVERVIEW (Unit: mm)



■ PINOUT TOP VIEW



■ BLOCK DIAGRAM



* Consists of 3 registers X, Y and Z

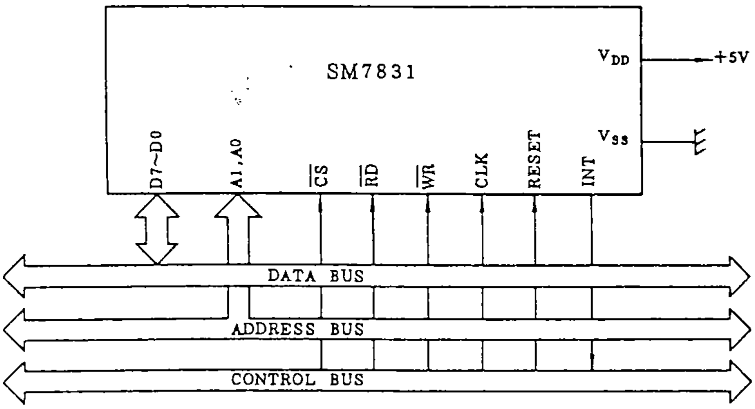
■ PIN DESCRIPTION

No.	Name	Description	No.	Name	Description
1	SFTx	Test pin	8	$\overline{WR}$	Write input. When the SM7831 is enabled by $\overline{CS}$, the data bus data is written into each register via D0 - D7 upon input of a negative logic signal to this pin.
2	CLK	External clock input			
3	INT	Operation end output signal This signal has positive logic and is output after the command input to the SM7831 has executed completely. This signal may be used as a CPU interrupt request signal. After input, the signal resets either when digit specification, command input, register input or reset is performed in the SM7831.	9	$\overline{CS}$	Chip select input This input enables the D0 to D7 pins of the SM7831 for communication between the data bus and the SM7831.
4	RESET	System reset input. The reset signal has positive logic and initializes the SM7831. The status register in the SM7831 is cleared upon reset.	10-17	D7-D0	Data bus bi-directional input/output Data and commands are transferred between the CPU and the SM7831 via these 8 input/output pins.
5, 6	A1, A0	Address bus inputs. Various SM7831 operations are selected using these two address control lines.	18	Vss	Ground
7	$\overline{RD}$	Read input. When the SM7831 is enabled by $\overline{CS}$, each register data is output to D0 - D7 upon input of a negative logic signal to this pin.	19 to 21	T9 to T7	Test pins
			22	V_{DD}	Power supply

■ COMMAND ALLOCATION TABLE

	Address		Data							
	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0
WR	0	0	Note) The number of arithmetic digits is specified by DSO and DS1.						DS1	DS0
	0	1	SL	SR	Y/X	←Arithmetic control command→				
	1	0	MEM ↓ MEM	DAP ↓ MEM	Y/X					
	1	1	← WRITE DATA →							
RD	0	0	BUSY				ERROR	SIGN	CARRY	ZERO
	0	1								
	1	0				DN4	DN3	DN2	DN1	DN0
	1	1	← READ DATA →							

■ SYSTEM CONFIGURATION



■ OPERATION TIME

The operation time is indicated by the unit time as shown in the table below. The actual time is obtained by multiplying the operation clock frequency to the unit time. In this table, n stands for the number of arithmetic digits, and m for the number of shift digits.

Item	Operation unit time	
	MIN	MAX
Clear		n + 7
Move		n + 7
Exchange		n + 7
Zero Sense		n + 7
Normalize		mn + 5m + 9
Shift Right		8m + 9
Shift Left		mn + 5m + 9
Add	n + 10	n + 14
Sub	n + 10	2n + 11
MLT	23n + 24	$5n^2 + 43n + 27$
DIV	$n^2 + 14n + 34$	$8n^2 + 47n + 24$
SQT	$n^2 + 14N + 26$	$12n^2 + 68n + 21$

■ OPERATION TIME EXAMPLE

14 digits, 2 MHz

Operation	MAX (μsec)	MIN (μsec)
Addition	14	12
Subtraction	20	12
Multiplication	805	173
Division	1125	213
Square root	1663	209

COMMAND LIST

R/W	Address		Command	Mnemonic	Processing	Data								Description
	A1	A0				D7	D6	D5	D4	D3	D2	D1	D0	
WR	0	0	Specification of the number of arithmetic digits	Mode Set	12 digits	0	0	0	0	0	0	0	0	Specify the number of arithmetic digits.
					14 digits	0	0	0	0	0	0	0	1	
					16 digits	0	0	0	0	0	0	1	0	
	0	1	Control command	Clear	X register clear	0	0	0	0	0	0	0	0	Clear all digits of the specified register (X,Y) to zero.
					Y register clear	0	0	1	0	0	0	0	0	
				Move	X → Y (X is unchanged)	0	0	0	0	0	0	0	1	Transfer the contents of the specified register (X,Y) to another register (X,Y). The contents of the first register remain unchanged.
					Y → X (Y is unchanged)	0	0	1	0	0	0	0	1	
				Exchange	X ⇄ Y	0	0	0	0	0	0	1	0	Exchange the contents of the specified register (Y,Z) with those of the X register. The contents of unspecified registers remain unchanged.
					X ⇄ Z	0	0	1	0	0	0	1	0	
				Zero Sence	X register 0 detection	0	0	0	0	0	0	1	1	Check if the specified register (X,Y) contains all zeros.
					Y register 0 detection	0	0	1	0	0	0	1	1	Zero Flag is set if the contents are zero.
				Normalize	X register normalization	0	0	0	0	0	1	0	0	When unnecessary zeros are in high-order digits of the specified register (X,Y), data is shifted leftward by the number of zero digits for normalization. After the shift, the number of shifted digits can be read from DN4 - DN0.
					Y register normalization	0	0	1	0	0	1	0	0	
				SL	X register shift left	1	0	0	S4	S3	S2	S1	S0	The shift operation can be performed by setting one of these bits to "1" and specifying the number of shift digits with 5 bits of D4 to D0 (binary). The arithmetic/ control command is executed when both SL (D7) and SR (D6) are "0".
					Y register shift left	1	0	1	S4	S3	S2	S1	S0	
				SR	X register shift right	0	0	0	S4	S3	S2	S1	S0	
					Y register shift right	0	1	1	S4	S3	S2	S1	S0	
	0	1	Arithmetic command	Add	X + Y → X	0	0	0	0	1	0	0	0	Adds the values of the X and Y registers and stores the result in the X register. When a carry is generated, 1-digit rightward shift is performed automatically, and Carry Flag is set. The contents of the Y register remain unchanged. Both the X and Y registers need not be normalized.
				Sub	X - Y → X	0	0	0	0	1	0	0	1	Subtract the Y register value from the X register value and store the result in the X register. Zero Flag is set when X = Y. Sign Flag is set when X < Y. The contents of the Y register remain unchanged. Both the X and Y registers need not be normalized.
				Mlt	X*Y → X	0	0	0	0	1	0	1	0	Multiply the X register value by the Y register value and store the result in the X register. If a carry is generated, the 1-digit right shift is carried out automatically, and Carry Flag is set. The contents of the Y register remain unchanged, and the contents of the X register remain unchanged in the Z register. Although the Y register need not be normalized, the X register must be normalized.
				Div	X/Y to X	0	0	0	0	1	1	0	0	Divide the X register value by the Y register value and store the result in the X register. The residual is stored in the Z register. Error Flag is set if this command is executed with the Y register set to zero. The contents of the Y register remain unchanged. Both the X and Y registers must be normalized.

R/W	Address		Command	Mnemonic	Processing	Data								Description
	A1	A0				D7	D6	D5	D4	D3	D2	D1	D0	
WR	0	1	Arithmetic command	SQRT$_{odd}$	$\sqrt{X_{odd}} \rightarrow X$	0	0	0	0	1	1	1	0	Calculate the square root of the X register and store the result in the X register. The X register value must be normalized. Note that the SQRT command differs depending on whether the number of digits before the decimal point is odd or even.
				SQRT$_{even}$	$\sqrt{X_{even}} \rightarrow X$	0	0	0	1	1	1	1	0	
	1	0	Data transfer	MEM → DAP	MEMORY → X register	1	0	0	0	0	0	0	0	Set this bit to "1" and specify the X and Y registers to enable MEM → DAP operation. Next, sequentially transfer the write data to store it in the specified register. Eight-bit data is transferred from the least significant digit (LSD) in units of 2 digits. For example, when the number of arithmetic digits is 16, D3 to D0 are transferred first as LSD and D7 to D4 are transferred as the
					MEMORY→ Y register	1	0	1	0	0	0	0	0	

R/W	Address		Command	Mnemonic	Processing	Data								Description
	A1	A0				D7	D6	D5	D4	D3	D2	D1	D0	
RD	0	0	Status flag	Status		Busy	-	-	-	Error	Sign	Carry	Zero	• Busy This bit goes "1" during operation. • Error This bit goes "1" when division is performed and the Y register is zero. • Sign This bit goes "1" when the result of the SUB command is X < Y (negative). • Carry This bit goes "1" when a carry is generated in the Add or Mlt command. • Zero This bit goes "1" when the X register is zero and Zero Sensor or another operation command is executed.
	1	0	Number of shift digits	Shift		-	-	-	*3 DN4	DN3	DN2	DN1	DN0	
	1	1	Read data	Read Data	Decimal, 2 digits	High-order digits				Low-order digits				

Notes

*1 S0 to S4 set the number of shift digits in binary.

*2 $\sqrt{X_{odd}}$: when the number of digits before the decimal point is odd in the X register.
$\sqrt{X_{even}}$: when the number of digits before the decimal point is even in the X register.

*3 DN0 to DN4 indicate the number of shift digits after the normalization command in binary.

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Rating	Unit	Remarks
Supply voltage	V_{DD} - V_{SS}	-0.3 to +6.5	V	
Input voltage	V_{IN}	V_{SS} -0.3 to V_{DD} +0.3	V	
Output voltage	V_{OUT}	V_{SS} -0.3 to V_{DD} +0.3	V	
Allowable dissipation	P_D	250	mW	Ta = 70°C
Storage temperature	T_{STG}	-40 to +125	°C	
Operating temperature	T_{OPR}	-20 to +70	°C	
Soldering temperature	T_{SLD}	260±5	°C	
Soldering time	t_{SLD}	10	Sec	

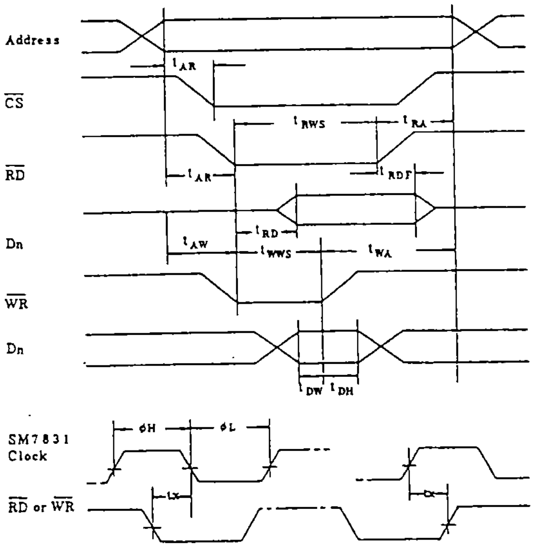
■ ELECTRICAL CHARACTERISTICS

<DC Characteristics>

V_{DD} = 5 V±5%, Ta = 0 to 70 °C unless otherwise noted.

ITEM	SYMBOL	CONDITIONS	LIMITS			UNIT
			MIN	TYP	MAX	
Current consumption	I_{DD}			10	25	mA
Operating voltage	V_{DD}		4.75	5.00	5.25	V
High-level output voltage	V_{OH}	I_{OH} = 400 µA other than test pin	3.5		5.25	V
Low-level output voltage	V_{OL}	I_{OL} = 2 mA other than test pin	0		0.4	V
High-level output voltage	V_{OH}	I_{OH} = 400 µA test pin	3.5		5.25	V
Low-level output voltage	V_{OL}	I_{OL} = 400 µA test pin	0		0.4	V
High-level input voltage	V_{IH}		2.4		5.25	V
Low-level input voltage	V_{IL}		0		0.8	V

<AC Characteristics>



* The "High" period of the SM7831 clock and the "Low" period of the $\overline{RD}$ or $\overline{WR}$ pulse must overlap by at least tx.

ITEM	SYMBOL	LIMITS		UNIT
		MIN	MAX	
Address -- $\overline{RD}$ or $\overline{CS}$ delay	t_{AR}	20		ns
$\overline{RD}$-Address delay	t_{RA}	20		ns
$\overline{RD}$ pulse width	t_{RWS}	400		ns
$\overline{RD}$-Data Bus Enable delay	t_{RD}		300	ns
$\overline{RD}$-Data Bus Disable delay	t_{RDF}		100	ns
Address-$\overline{RD}$ delay	t_{AW}	20		ns
$\overline{WR}$-Address delay	t_{WA}	20		ns
$\overline{WR}$ pulse width	t_{WWS}	400		ns
$\overline{WR}$-Data Bus Enable overlap	t_{DW}	50		ns
$\overline{WR}$-Data Bus Disable overlap	t_{DH}	50		ns
Operating frequency	f_{OPG}	DC	2.0	MHz
SM7831 clock pulse width	øH	200		ns
(tr, tf ≤20ns)	øL	300		ns
Clock-$\overline{RD}$,$\overline{WR}$ overlap	tx	100		ns

■ EXPLANATION OF APPLICATION

The following describes an example of executing various arithmetic operations with 14-digit decimal floating-point data. In this example, the 8085A is used for CP. In the flowchart, parts enclosed in parenthesis (▭) are executed by the SM7831, and the rest by the CPU.

1. Data format

Format of 14-digit decimal floating-point data

Memory address	B7	B6	B5	B4	B3	B2	B1	B0
0	Mantissa sign	Format of 14-digit decimal floating-point data						
1						LSD		
2								
3								
4								
5								
6								
7		MSD						

Note 1. The mantissa part is always normalized. Decimal number only.

Note 2 Sign of the mantissa
"0" ... Positive
"1" ... Negative

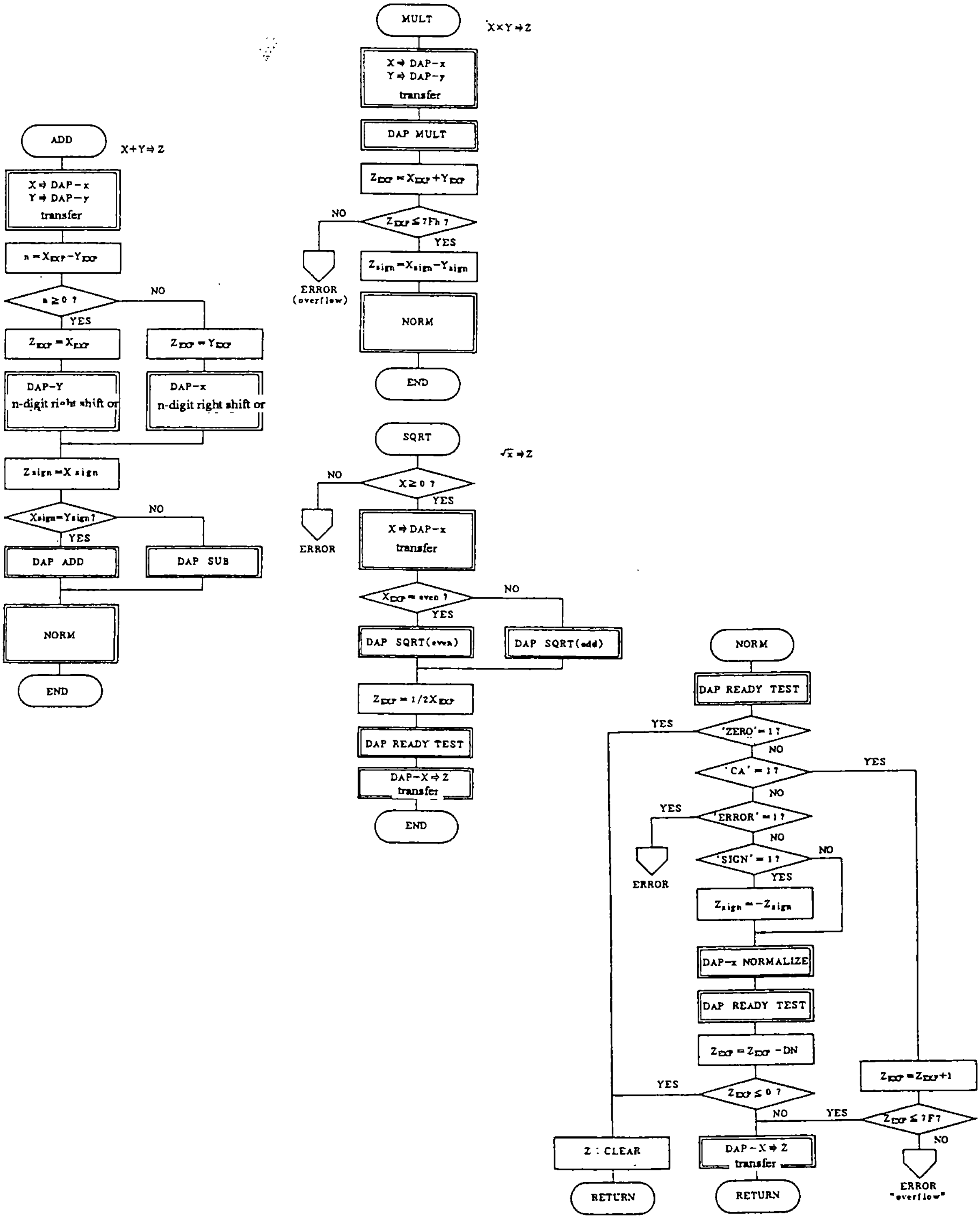
Note 3 Exponent data example

Exponent data	Contents
00	When data is "0"
01	10^{-63}
02	10^{-62}
·	·
3FH	10^{-1}
40h	10^{0}
41h	10^{1}
·	·
7Fh	10^{63}

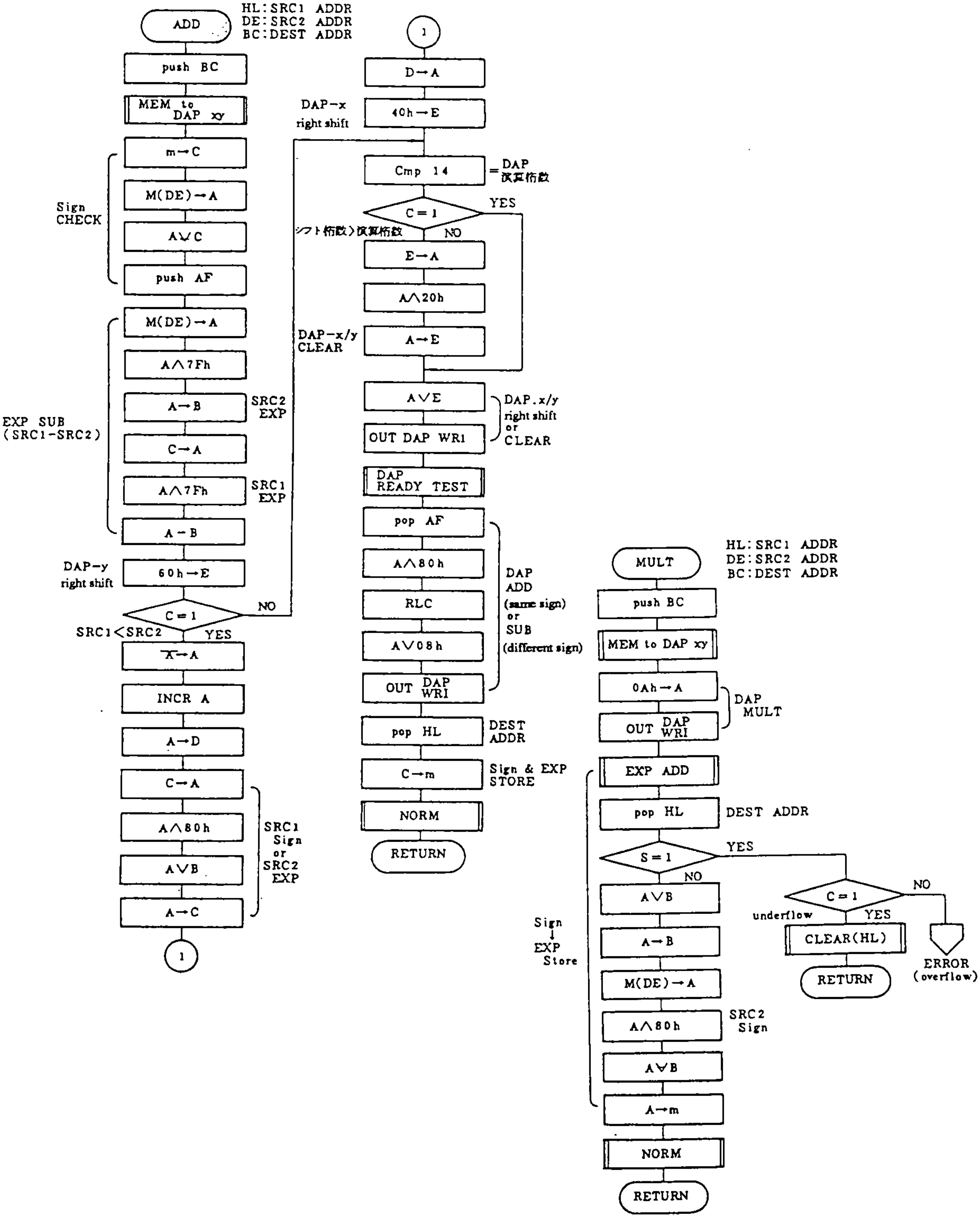
[Example] How to represent 123.45

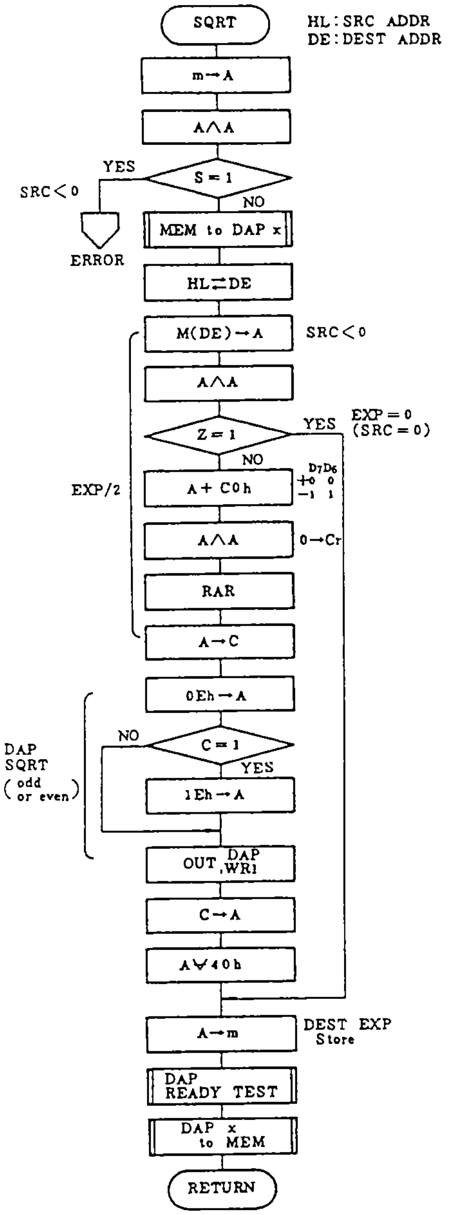
Memory address		
0	4	2
1	0	0
2	0	0
3	0	0
4	0	0
5	5	0
6	3	4
7	1	2

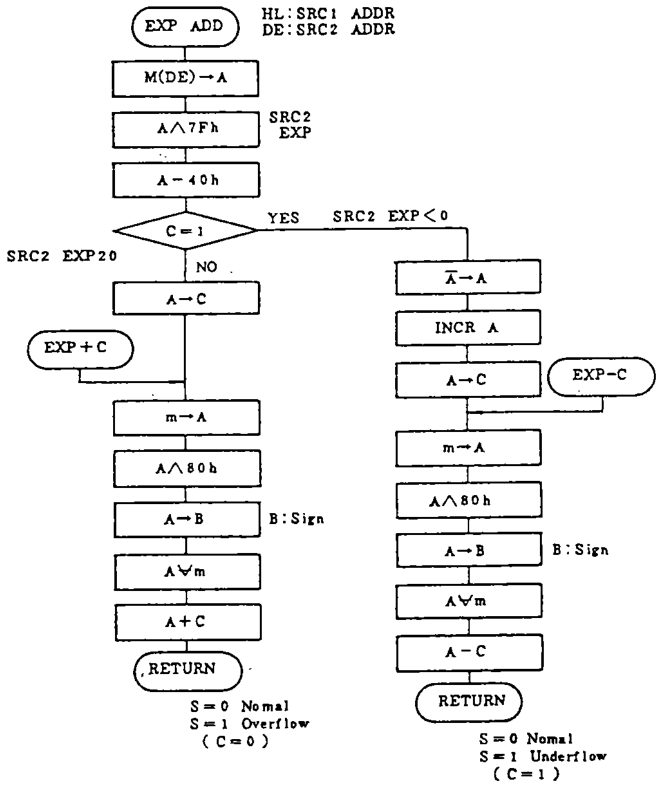
2. Flowchart

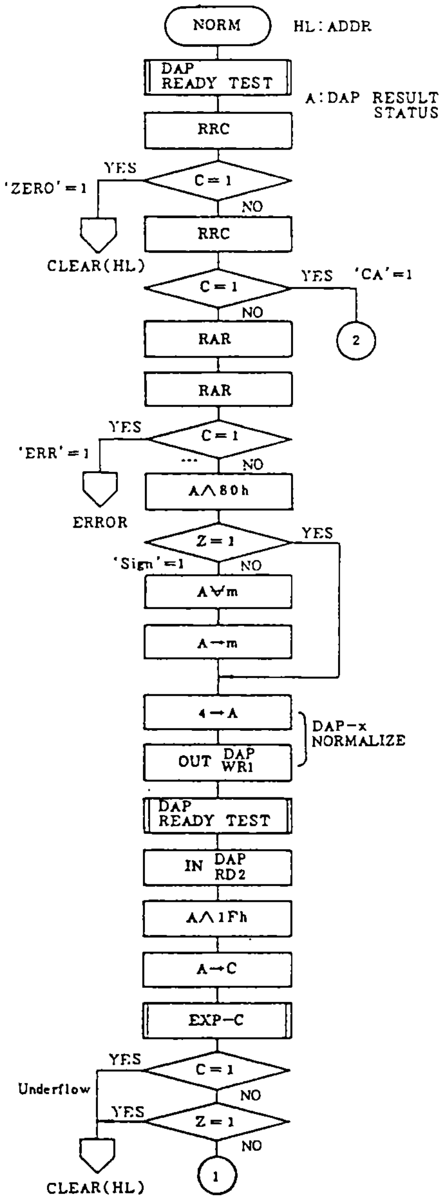

MULT
X×Y⇒Z
X⇒DAP-x
Y⇒DAP-y
transfer
DAP MULT
$Z_{EXP} = X_{EXP} + Y_{EXP}$
$Z_{EXP} \leq 7Fh$?
NO
YES
ERROR
(overflow)
$Z_{sign} = X_{sign} - Y_{sign}$
NORM
END
ADD
X+Y⇒Z
X⇒DAP-x
Y⇒DAP-y
transfer
$n = X_{EXP} - Y_{EXP}$
n ≥ 0 ?
NO
YES
$Z_{EXP} = X_{EXP}$
$Z_{EXP} = Y_{EXP}$
DAP-Y
n-digit right shift or
DAP-x
n-digit right shift or
$Z_{sign} = X_{sign}$
$X_{sign} = Y_{sign}$?
NO
YES
DAP ADD
DAP SUB
NORM
END
SQRT
√x⇒Z
X ≥ 0 ?
NO
YES
ERROR
X⇒DAP-x
transfer
X_{EXP} = even ?
NO
YES
DAP SQRT(even)
DAP SQRT(odd)
$Z_{EXP} = 1/2X_{EXP}$
DAP READY TEST
DAP-X⇒Z
transfer
END
NORM
DAP READY TEST
'ZERO'=1 ?
YES
NO
'CA'=1 ?
YES
NO
'ERROR'=1 ?
YES
NO
ERROR
'SIGN'=1 ?
NO
YES
$Z_{sign} = -Z_{sign}$
DAP-x NORMALIZE
DAP READY TEST
$Z_{EXP} = Z_{EXP} - DN$
$Z_{EXP} \leq 0$?
YES
NO
$Z_{EXP} = Z_{EXP} + 1$
$Z_{EXP} \leq 7F$?
YES
NO
ERROR
"overflow"
Z : CLEAR
RETURN
DAP-X⇒Z
transfer
RETURN

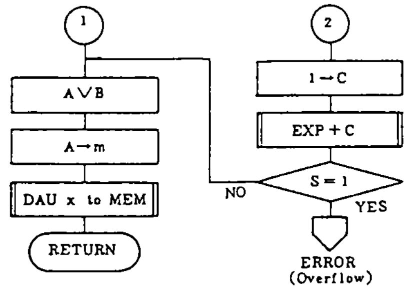
3. 8085 program example

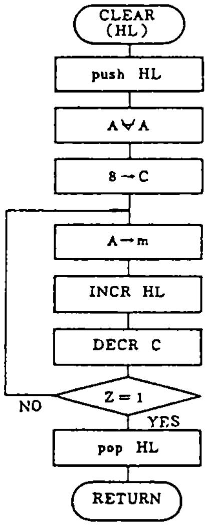



SQRT
HL:SRC ADDR
DE:DEST ADDR
m→A
A∧A
S=1
YES
SRC<0
ERROR
NO
MEM to DAP x
HL⇄DE
M(DE)→A
SRC<0
A∧A
Z=1
YES
EXP=0
(SRC=0)
NO
EXP/2
A+C0h
D7D6
+0 0
−1 1
A∧A
0→Cr
RAR
A→C
0Eh→A
DAP
SQRT
(odd or even)
NO
C=1
YES
1Eh→A
OUT DAP ,WR1
C→A
A∀40h
A→m
DEST EXP
Store
DAP
READY TEST
DAP x
to MEM
RETURN


EXP ADD
HL:SRC1 ADDR
DE:SRC2 ADDR
M(DE)→A
A∧7Fh
SRC2
EXP
A−40h
C=1
YES
SRC2 EXP<0
SRC2 EXP≥0
NO
A→C
EXP+C
m→A
A∧80h
A→B
B:Sign
A∀m
A+C
RETURN
S=0 Nomal
S=1 Overflow
(C=0)
Ā→A
INCR A
A→C
EXP−C
m→A
A∧80h
A→B
B:Sign
A∀m
A−C
RETURN
S=0 Nomal
S=1 Underflow
(C=1)





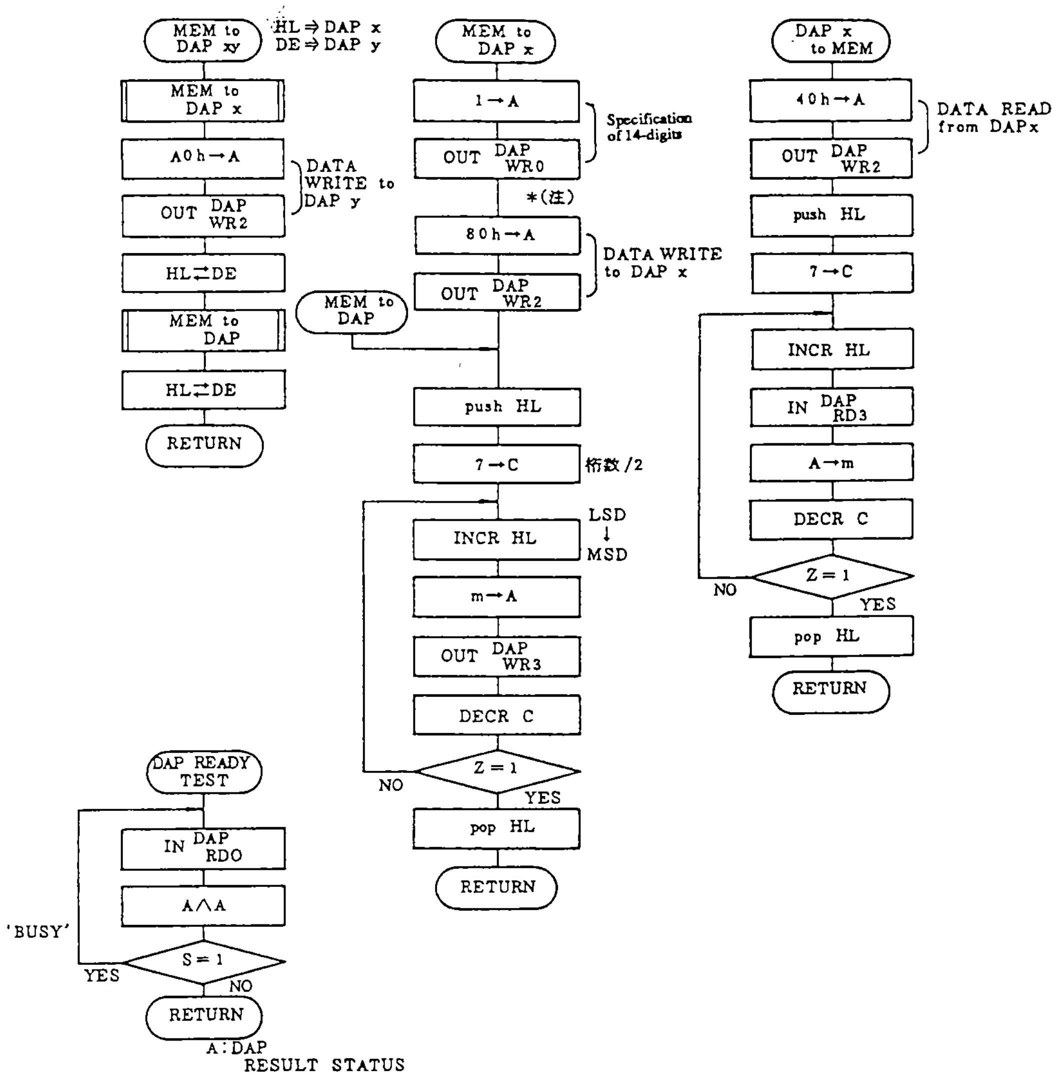


- NORM routine common to ADD, SUB, MULT and DIV.

  The dedicated NORM routine must consider only the following STATUS.

  'ZERO' ----- common
  'CA' -------- ADD, MULT
  'ERR' -------- DIV
  'SIGN' ------- SUB

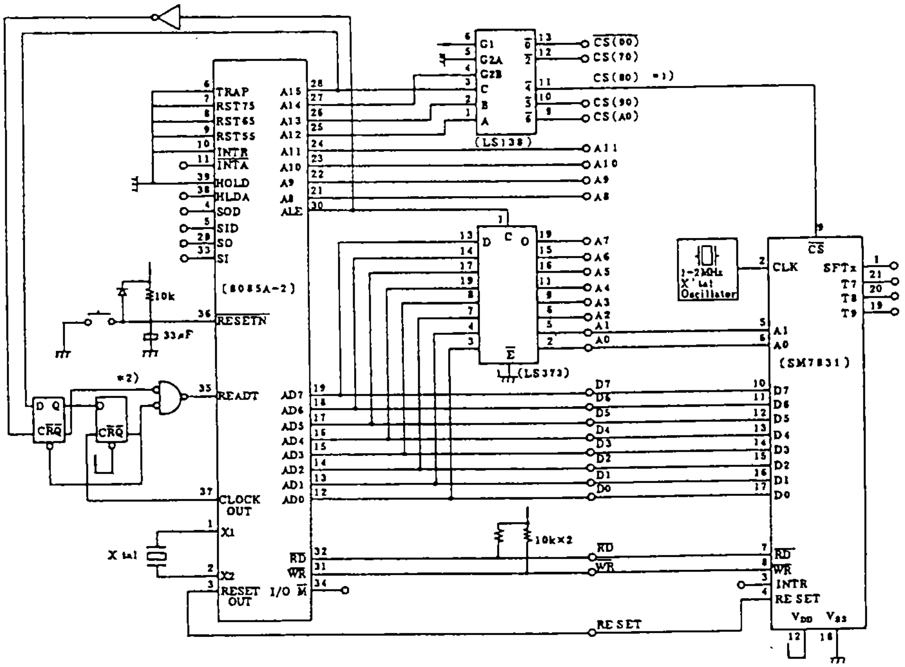
The routine below DAP-x, NORMALIZE is needed for SUB and DIV. It can be omitted for ADD and MULT.



(Note) Though the specification of the number of digits is included in this routine, it must be performed only once in a routine of system initialization etc.

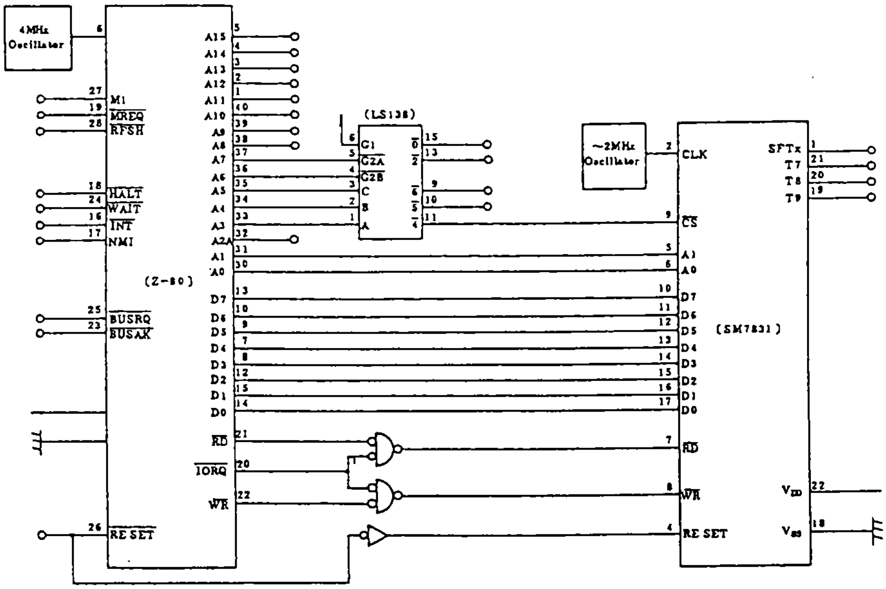
■ TYPICAL APPLICATIONS

1. Connection with the 8085A (reference circuit)



*1 I/O port address is in the order of 8000.
*2 Wait Control example

2. Connection with the Z-80 (reference circuit)



■ SUPPLEMENTARY EXPLANATION

Source list of arithmetic operations (addition, subtraction, multiplication, division, square root) performed by the 8088

In this example, the data format is set as shown below.

01h	− 127
02h	− 126
⋮	⋮
7Fh	− 1
80h	0
81h	+ 1
⋮	⋮
FFh	+ 127

<15-digit floating point data>

Memory address		
0	Exponent data (hexadecimal)	
1	MSD	MSD-1
2	MSD-2	
3		
4		
5		
6		
7	LSD+2	LSD+1
8	LSD	Mantissa sign

Note 1: Exponent data is shown below.
When mantissa data is "0", the exponent is set to "0".

Note 2: Signs are as follows.
When mantissa data is positive ... "Ch"
When mantissa data is negative ..."Dh"

```
LOC   OBJ             LINE    SOURCE

                              ;
                              ;******************************************
                              ;*        Function_Module  (2)             *
                              ;******************************************
                              ;
                              NAME      FUNC2
                              ;
                              ;
                              ;                  ....     '83. 09. 13
                              ;                  ....         09. 22
                              ;
                              ;
                              ;
                              ;
                              PUBLIC             Addition, Additon_W,
                              &                  Subtraction, Subtraction_W,
                              &                  Multiply, Multiply_W,
                              &                  Division, Division_W,
                              &                  Square_Root, Square_Root_W,
                              &                  Integer, Integer_W,
                              &                  Round_UP, Round_UP_W,
                              &                  Round, Round_W,
                              &                  Round_Down, Round_Down_W,
                              &                  Clear_W
                              ;
                              ;
                              ;
                              Function_Module       SEGMENT PUBLIC 'CODE'
                              ;
                              ASSUME             CS:Function_Module
                              ;
                              ;
                              EXTRN              WORK_0:byte, WORK_1:byte, WORK_2:byte, WORK_3:byte,
                              &                  WORK_4:byte, WORK_5:byte, WORK_6:byte, WORK_7:byte
                              ;
                              EXTRN              Func_error:near
                              &
                              ;
0000                          Function_Sub1         PROC
                              ;
                              ;
                              ;    ********************************
                              ;    *        Function_Sub  (1)     *
                              ;    ********************************
                              ;
  0030                                DAU_DIGIT         EQU      30H
  0032                                DAU_COMMAND       EQU      32H
  0034                                DAU_DIRECTION     EQU      34H
  0036                                DAU_WRITE_DATA    EQU      36H
                               ;
    0030                              DAU_STATUS        EQU      30H
    0034                              DAU_NORM_STATUS   EQU      34H
```

```
LOC   OBJ          SOURCE

0036                         DAU_READ_DATA   EQU     36H
                   ;
                   ;
                   ;*****  Addition (W)  *****
                   ;                (DS:SI) + (ES:DI) => (DS:SI)
0000               Addition_W:
0000  8BDE                   MOV     BX, SI
                   ;
                   ;*****  Addition  *****
                   ;                (DS:SI) + (ES:DI) => (DS:BX)
0002               Addition:
0002  32D2                   XOR     DL, DL
0004  EB04                   JMP     short ADD_or_SUB
                   ;
                   ;
                   ;*****  Subtraction (W)  *****
                   ;                (DS:SI) - (ES:DI) => (DS:SI)
0006               Subtraction_W:
0006  8BDE                   MOV     BX, SI
                   ;
                   ;*****  Subtraction  *****
                   ;                (DS:SI) - (ES:DI) => (DS:BX)
0008               Subtraction:
0008  B201                   MOV     DL, 1
                   ;...       JMP     short ADD_or_SUB
                   ;
                   ;*****  ADD_or_SUB  ****
                   ;
000A               ADD_or_SUB:
000A  325408                 XOR     DL, DS:[SI+8]
000D  26325508               XOR     DL, ES:[DI+8]
0011  80E201                 AND     DL, 1
                   ;
0014               ADD_or_SUB_2:
0014  8A04                   MOV     AL, DS:[SI]
0016  268A25                 MOV     AH, ES:[DI]
0019  B620                   MOV     DH, 20H
001B  3AC4                   CMP     AL, AH
001D  7304                   JNC     J_001
001F  86E0                   XCHG    AH, AL
0021  32F6                   XOR     DH, DH
0023                         J_001:
0023  8807                   MOV     [BX], AL
0025  2AC4                   SUB     AL, AH                 ; AL = GRATE EXP
0027  3C0F                   CMP     AL, 0FH
0029  7304                   JNC     J_002
002B  0C40                   OR      AL, 40H
002D  0AF0                   OR      DH, AL
002F                         J_002:                                 : X/Y   R_SHIFT
                   ;
002F  E8DA00                 CALL    MEM_to_DAU_XY
                   ;
0032  8AC6                   MOV     AL, DH
0034  E632                   OUT     DAU_COMMAND, AL
                   ;
0036  E82501                 CALL    DAU_READY_TEST
0039  8A4408                 MOV     AL, DS:[SI+8]
003C  884708                 MOV     [BX+8], AL
                   ;
003F  8AC2                   MOV     AL, DL
0041  0408                   ADD     AL, 8                          : (DL)  0/1
0043  E632                   OUT     DAU_COMMAND, AL                ; 8/9   ADD/SUB
0045  7A13                   JP      NORM_W_2
                   ;...       JMP     short NORM_W_1
                   ;
                   ; ****    NORM W  (1)   ****
                   ;
0047               NORM_W_1:
0047  E81401                 CALL    DAU_READY_TEST

004A  D0E8                   SHR     AL, 1                  ; DAU 'ZERO'?
004C  7229                   JC      Clear_W

004E  D0E8                   SHR     AL, 1                  ; DAU 'CA'?
0050  7304                   JNC     J_101

0052  FE07                   INC     byte ptr [BX]          ; DAU 'CA'-1
0054  7448                   JZ      ENZAN_ERROR
0056               J_101:
0056  E8E300                 CALL    DAU_X_to_MEM
0059  C3                     RET
```

```
LOC  OBJ             LINE   SOURCE

                      134   ;
                      135   ; ****  NORM W  (2)  ****
                      136   ;
005A                  137   NORM_W_2:
005A E80101           138              CALL     DAU_READY_TEST
                      139                                          ; DAU 'ZERO'?
005D D0E8             140              SHR      AL, 1
005F 7216             141              JC       Clear_W
                      142   ;
0061 D0E8             143              SHR      AL, 1
0063 2401             144              AND      AL, 1
0065 304708           145              XOR      [BX+8], AL         ; 'SIGN'=1  CHG_SIGN
                      146                                          ; DAU X_NORMALIZE
0068 B004             147              MOV      AL, 4
006A E632             148              OUT      DAU_COMMAND, AL
                      149                                          ;
006C E8EF00           150              CALL     DAU_READY_TEST
                      151                                          ;
006F E434             152              IN       AL, DAU_NORM_STATUS
0071 241F             153              AND      AL, 1FH
0073 2807             154              SUB      [BX], AL
0075 77DF             155              JA       J_101
                      156   ;...       JMP      short Clear_W
                      157   ;
                      158   ;
                      159   ; ****' Clear W    ****
                      160   ;                   (DS:BX)
0077                  161   Clear_W:
0077 1E               162              PUSH     DS
0078 07               163              POP      ES
                      164              ;
0079 57               165              PUSH     DI
007A 8BFB             166              MOV      DI, BX
007C B90400           167              MOV      CX, 4
007F 33C0             168              XOR      AX, AX
0081 FC               169              CLD
0082 F3               170         REP  STOSW
0083 AB
0084 C6050C           171              MOV      byte ptr[DI], 0CH
0087 5F               172              POP      DI
0088 C3               173              RET
                      174   ;
                      175   ;
                      176 +1 $EJECT
                      177   ;
                      178   ;
                      179   ;*****  Multiply (W)   *****
                      180   ;                   (DS:SI) * (ES:DI) -> (DS:SI)
0089                  181   Multiply_W:
0089 8BDE             182              MOV       BX, SI
                      183   ;
                      184   ;*****  Multiply     *****
                      185   ;                   (DS:SI) * (ES:DI) -> (DS:BX)
008B                  186   Multiply:
008B E87E00           187              CALL     MEM_to_DAU_XY
                      188                                          ; DAU X*Y
008E B00A             189              MOV      AL, 0AH
0090 E632             190              OUT      DAU_COMMAND, AL
                      191                                          ;
0092 E82300           192              CALL     SIGN_EXP_STORE1
0095 71B0             193              JNO      NORM_W_1
                      194   ;
0097                  195    OVERFLOW_CHECK:
0097 9C               196              PUSHF
0098 E8C300           197              CALL     DAU_READY_TEST
009B 9D               198              POPF
009C 72D9             199              JC       Clear_W
                      200   ;
009E                  201   ENZAN_ERROR:
009E E90000      E    202              JMP      Func_Error
```

```
LOC   OBJ           SOURCE

                    ;
                    ;
                    ;*****  Division (W)   *****
                    ;                 (DS:SI) / (ES:DI) -> (DS:SI)
00A1                Division_W:
00A1  8BDE                  MOV     BX, SI
                    ;
                    ;*****  Division   *****
                    ;                 (DS:SI) / (ES:DI) -> (DS:BX)
00A3                Division:
00A3  26F64501FF            TEST    byte ptr ES:[DI+1], 0FFH
00A8  74F4                  JZ      ENZAN_ERROR        ;  (ES:DI) =0
                    ;
00AA  E85F00                CALL    MEM_to_DAU_XY
                                                       ; DAU X/Y
00AD  B00C                  MOV     AL, 0CH
00AF  E632                  OUT     DAU_COMMAND, AL
                                                       ;
00B1  E80C00                CALL    SIGN_EXP_STORE2
00B4  71A4                  JNO     NORM_W_2
00B6  EBDF                  JMP     OVERFLOW_CHECK
                    ;
                    ;
                    ; ****  SIGN , EXP STORE (1)   ****
                    ;               SIGN (XOR) , EXP (ADD)
00B8                SIGN_EXP_STORE1:
00B8  268A25                MOV     AH, ES [DI]
00BB  80EC80                SUB     AH, 80H
00BE  EB05                  JMP     short J_121
                    ;
                    ; ****  SIGN , EXP STORE (2)   ****
                    ;               SIGN (XOR) , EXP (SUB)
00C0                SIGN_EXP_STORE2:
00C0  B480                  MOV     AH, 80H
00C2  262A25                SUB     AH, ES: [DI]
00C5                J_121:
00C5  8A4408                MOV     AL, [SI+8]
00C8  26324503              XOR     AL, ES: [DI+8]
00CC  0C0C                  OR      AL, 00001100B
00CE  884708                MOV     [BX+8], AL

00D1  8A04                  MOV     AL, [SI]
00D3  2C80                  SUB     AL, 80H
00D5  02C4                  ADD     AL, AH
00D7  7004                  JO      J_122
00D9  3480                  XOR     AL, 80H
00DB  8807                  MOV     [BX], AL
00DD                                J_122:
00DD  C3                    RET
                    ;
                    ;
                    ;*****  Square_Root (W)   *****
                    ;               Square_Root (DS:SI) -> (DS:SI)
00DE                Square_Root_W:
00DE  8BDE                  MOV     BX, SI
                    ;
                    ;*****  Square_Root   *****
                    ;               Square_Root (DS:SI) -> (DS:BX)
00E0                Square_Root:
00E0  F6440801              TEST    byte ptr [SI+8], 1
00E4  75B8                  JNZ     ENZAN_ERROR        ;  (DS:SI) <0
                                                       ;
00E6  E83600                CALL    MEM_to_DAU_X

00E9  8A24                  MOV     AH, [SI]
00EB  22E4                  AND     AH, AH
00ED  7414                  JZ      J_123
00EF  80F480                XOR     AH, 80H
00F2  32C0                  XOR     AL, AL
00F4  D1F8                  SAR     AX, 1
00F6  80F480                XOR     AH, 80H
                                                       ; DAU X SORT
00F9  D0E8                  SHR     AL, 1
00FB  D0E8                  SHR     AL, 1
00FD  D0E8                  SHR     AL, 1
00FF  0C0E                  OR      AL, 0EH            ; AL = 0E/1E
0101  E632                  OUT     DAU_COMMAND, AL
0103                        J_123:                     ; EXP STORE
0103  8827                  MOV     [BX], AH
                                                       ;
0105  E85600                CALL    DAU_READY_TEST
0108  E83100                CALL    DAU_X_to_MEM
010B  C3                    RET
```

```
LOC   OBJ             LINE     SOURCE

                       287     ;
                       288 +1  $EJECT
                       289     ;
                       290     ;
                       291     ;  ****   MEM  to  DAU  (X, Y)    ****
                       292     ;                   (DS:SI) ->DAUX, (ES:DI) ->DAUY
010C                   293     MEM_to_DAU_XY:
010C  E81000           294               CALL      MEM_to_DAU_X
                       295     ;
010F                   296     MEM_to_DAU_Y:
010F  1E               297               PUSH      DS
0110  8CC1             298               MOV       CX, ES                    ;
0112  8ED9             299               MOV       DS, CX                    ;  ES -> DS
0114  87F7             300               XCHG      SI, DI
                       301                                                   ;
0116  B0A0             302               MOV       AL, 0A0H                  ;  DAU WRITE Y
0118  E80C00           303               CALL      MEM_to_DAU
                       304                                                   ;
011B  87F7             305               XCHG      SI, DI
011D  1F               306               POP       DS
011E  C3               307               RET
                       308     ;
                       309     ;
                       310     ;  ****   MEN  to  DAU  (X)    ****
                       311     ;                   (DS:SI) ->DAUX
011F                   312     MEM_to_DAU_X:
011F  B002             313               MOV          AL, 2                  ;  DAU 16DIGIT
0121  E630             314               OUT       DAU_DIGIT, AL
                       315     ;
0123  90               316               NOP                                 ;  DELAY FOR DAU_7031
0124  90               317               NOP                                 ;
                       318     ;
                       319                                                   ;
0125  B080             320               MOV       AL, 80H                   ;  DAU WRITE X
0127                   321     MEM_to_DAU:
0127  E634             322               OUT       DAU_DIRECTION, AL
                       323               ;
0129  83C608           324               ADD       SI, 8
012C  FD               325               STD
012D  AC               326               LODSB
012E  24F0             327               AND       AL, 0F0H
0130  E636             328               OUT       DAU_WRITE_DATA, AL
                       329               ;
0132  B90700           330               MOV       CX, 7
                       331               ;
0135                   332               J_131:
0135  AC               333                LODSB
0136  E636             334                OUT        DAU_WRITE_DATA, AL
0138  E2FB             335               LOOP      J_131
                       336               ;
013A  FC               337               CLD
013B  C3               338               RET
                       339     ;
                       340     ;
                       341     ;  ****   DAU  (X)  to  MEM   ****
                       342     ;                   DAU (X) -> (DS:BX)
013C                   343     DAU_X_to_MEM:
013C  1E               344               PUSH      DS
013D  07               345               POP       ES
013E  87FB             346               XCHG      DI, BX
                       347               ;
0140  B040             348               MOV       AL, 40H
0142  E634             349               OUT       DAU_DIRECTION, AL      ;  DAU READ X
                       350               ;
0144  83C708           351               ADD       DI, 8
0147  80250F           352               AND       byte ptr [DI], 0FH
014A  E436             353               IN        AL, DAU_READ_DATA
014C  24F0             354               AND       AL, 0F0H
014E  0805             355               OR        [DI], AL
0150  4F               356               REC       DI
                       357                                                   ;
0151  B90700           358               MOV       CX, 7
0154  FD               359               STD
                       360               ;
0155                   361               J_133:
0155  E436             362                IN        ALL, DAU_READ_DATA
0157  AA               363                STOSB
0158  E2FB             364               LOOP      J_133
                       365               ;
015A  FC               366               CLD
015B  87FB             367               XCHG      DI, BX
015D  C3               368               RET
```

```
LOC   OBJ             LINE     SOURCE

                       369     ;
                       370     ;
                       371     ; ****   DAU READY TEST   ****
                       372     ;
015E                   373     DAU_READY_TEST:
015E  E430             374               IN        AL, DAU_STATUS
0160  22C0             375               AND       AL, AL
0162  78FA             376               JS        DAU_READY_TEST
0164  C3               377               RET
                       378     ;
                       379     ;
                       380     ;
                       381  +1 $EJECT
```