Mk14 Micro Computer

program

TITLE / TITRE: Square root

Square root of a 2-byte number (maximum 65535).

*Application of the example from the manual “**MK14 MicroComputer Training Manual**” page 71*

1) Input the number :
   go to address 0F20
   then input the number xxxx in two parts
   [Abort]0F20[Term]xx[MEM]xx[MEM]

2) Launch :
   [Abort]0F24[GO]

3) Result :
   [Abort]0F21[Term] display the result

```
:100F200000000000C400C8FB03B8F8F0F601C4FEDE
:100F3000F40001F0EDC8EB40F0E7C8E51D94029025
:100F4000E7C400C8DCF8DCC8D93F0000000000009E
:00000001FF
```

```
0f24             .OR    0xf24

    0f20    NbrH    .eq 0xf20
    0f21    NbrL    .eq 0xf21
    0f22    Calc    .eq 0xf22

0f24        SQR:
0f24 c4 00    LDI 0x00
0f26 c8 fb    ST Calc

0f28        BCL:
0f28 03       SCL
0f29 b8 f8    DLD Calc
0f2b f0 f6    ADD Calc
0f2d 01       XAE
0f2e c4 fe    LDI 0xFE
0f30 f4 00    ADI 0x00
0f32 01       XAE
0f33 f0 ed    ADD NbrL
0f35 c8 eb    ST NbrL
0f37 40       LDE
0f38 f0 e7    ADD NbrH
0f3a c8 e5    ST NbrH
0f3c 1d       SRL
0f3d 94 02    JP FIN
0f3f 90 e7    JMP BCL
0f41        FIN:
0f41 c4 00    LDI 0x00
0f43 c8 dc    ST NbrH
0f45 f8 dc    CAD Calc
0f47 c8 d9    ST NbrL
0f49 3f       XPPC 3
```

Mk14 Micro Computer

program

TITLE / TITRE: Factorial (Version 2)

PAGE 1 OF 1

Calculates the factorial of n (n stored in register 0f20)
Limited to 8! Result is **16 bits**, therefore a maximum of FFFF. *(65535 in **decimal**)*

Example : 8!

1) Input 8 at address **0f20** :	**[Abort]0F20[Term]08[Mem]**
2) Launch the program :	**[Abort]0F27[GO]**
3) Result at addresses **0f25** and **0f26** :	**[Abort]0F24[Term] ... [Mem]**
display :	**9d80** *(40320 in **decimal**)*

```
0f27            .OR    0xf27
    0f20     Nbr     .eq 0xf20
    0f21     Mul     .eq 0xf21
    0f22     Cpt     .eq 0xf22
    0f23     Nbr1H   .eq 0xf23
    0f24     Nbr1L   .eq 0xf24
    0f25     Nbr2H   .eq 0xf25
    0f26     Nbr2L   .eq 0xf26
0f27         FAC:
0f27 c0 f8     LD Nbr
0f29 c8 f7     ST Mul
0f2b c8 fa     ST Nbr2L
0f2d c4 00     LDI 0
0f2f c8 f3     ST Nbr1H
0f31 c8 f3     ST Nbr2H
0f33 c8 f0     ST Nbr1l
0f35 90 0e     JMP BC1
0f37         BC0:
0f37 c0 eb     LD Nbr1H
0f39 c8 eb     ST Nbr2H
0f3b c0 e8     LD Nbr1L
0f3d c8 e8     ST Nbr2L
0f3f c4 00     LDI 0
0f41 c8 e1     ST Nbr1H
0f43 c8 e0     ST Nbr1L
```

```
0f45         BC1:
0f45 c0 db     LD Mul
0f47 03        SCL
0f48 fc 01     CAI 1
0f4a c8 d6     ST Mul
0f4c c8 d5     ST Cpt
0f4e         BC2:
0f4e 02        CCL
0f4f c0 d4     LD Nbr1L
0f51 f0 d4     ADD Nbr2L
0f53 c8 d0     ST Nbr1L
0f55 c0 cd     LD Nbr1H
0f57 f0 cd     ADD Nbr2H
0f59 c8 c9     ST Nbr1H
0f5b b8 c6     DLD Cpt
0f5d 9c ef     JNZ BC2
0f5f b8 c0     DLD Nbr
0f61 9c d4     JNZ BC0
0f63 3f        XPPC 3
```


:100F2000000000000000C0F8C8F7C8FAC400C8FC
:100F3000F3C8F3C8F0900EC0EBC8EBC0E8C8E8C433
:100F400000C8E1C8E0C0DB03FC01C8D6C8D502C0B8
:100F5000D4F0D4C8D0C0CDF0CDC8C9B8C69CEFB8C5
:100F6000C09CD43F000000000000000000000000012
:00000001FF

Calculates the factorial of n (n stored in register 0f20)
Limited to 5! Result is 8 bits, therefore a maximum of FF. *(255 in **decimal**)*

Example : 5!
1) Input 5 at address **0f20** : **[Abort]0F20[Term]05[Mem]**
2) Launch the program : **[Abort]0F25[GO]**
3) Result at address **0f24** : **[Abort]0F24[Term]**
display : **78** *(120 in **decimal**)*

```
:100F2000000000000C0FAC8F9C8FAC400C8F49074
:100F300008C0F0C8F0C400C8EAC0E703FC01C8E27A
:100F4000C8E2C0DF02F0DEC8DAB8D99CF5B8D29C9E
:100F5000E03F0000000000000000000000000072
:00000001FF
```

```
0f25              .OR    0xf25
    0f20    Nbr    .eq 0xf20
    0f21    Mul    .eq 0xf21
    0f22    Cal1   .eq 0xf22
    0f23    Cal2   .eq 0xf23
    0f24    Fact   .eq 0xf24
0f25        FAC:
0f25 c0 fa     LD Nbr
0f27 c8 f9     ST Mul
0f29 c8 fa     ST Fact
0f2b c4 00     LDI 0
0f2d c8 f4     ST Cal1
0f2f 90 08     JMP BC1
0f31        BC0:
0f31 c0 f0     LD Cal1
0f33 c8 f0     ST Fact
0f35 c4 00     LDI 0
0f37 c8 ea     ST Cal1
0f39        BC1:
0f39 c0 e7     LD Mul
0f3b 03        SCL
0f3c fc 01     CAI 1
0f3e c8 e2     ST Mul
0f40 c8 e2     ST Cal2
0f42        BC2:
0f42 c0 df     LD Cal1
0f44 02        CCL
0f45 f0 de     ADD Fact
0f47 c8 da     ST Cal1
0f49 b8 d9     DLD Cal2
0f4b 9c f5     JNZ BC2
0f4d b8 d2     DLD Nbr
0f4f 9c e0     JNZ BC0
0f51 3f        XPPC 3
```

Mk14 Micro Computer **program**

TITLE / TITRE: Fibonacci PAGE / PAGE 1 OF / DE 1

Calculate the nth Fibonacci number (stored at address 0F20)
(n is limited to 13 since 8-bit memory registers are limited to FF)

1) Input :
input of rank nn at address 0F20 (in hex)
[Abort]0F20[Term]nn[MEM]

2) Launch :
[Abort]0F25[GO]

3) Result :
[Abort]0F21[Term] display the result (in hex)

```
0f25                .OR  0xf25
       0f20    Nbr  .eq 0xf20
       0f21    Fibo .eq 0xf21
       0f22    Clc1 .eq 0xf22
       0f23    Clc2 .eq 0xf23
       0f24    Decr .eq 0xf24

0f25 c4 01   FIB: LDI 1
0f27 c8 f9        ST Fibo
0f29 c4 00        LDI 0
0f2b c8 f6        ST Clc1
0f2d c0 f2        LD Nbr
0f2f c8 f4        ST Decr
0f31 b8 f2        DLD Decr
0f33 02      LBL: CCL
0f34 c0 ec        LD Fibo
0f36 f0 eb        ADD Clc1
0f38 c8 ea        ST Clc2
0f3a c0 e6        LD Fibo
0f3c c8 e5        ST Clc1
0f3e c0 e4        LD Clc2
0f40 c8 e0        ST Fibo
0f42 b8 e1        DLD Decr
0f44 9c ed        JNZ LBL
0f46 c4 00        LDI 0
0f48 c8 d9        ST Clc1
0f4a c8 d8        ST Clc2
0f4c 3f           XPPC 3
```


```
:100F20000000000000C401C8F9C400C8F6C0F2C83F
:100F3000F4B8F202C0ECF0EBC8EAC0E6C8E5C0E4E1
:100F4000C8E0B8E19CEDC400C8D9C8D83F00000093
:00000001FF
```

Calculate the Greatest Common Divisor between two numbers n1 and n2

(n1 and n2 are limited to 255 because 8-bit memory registers are limited to FF)

1) Numbers entry :

go to address 0F20

then input first number xx then input second number yy

`[Abort]0F20[Term]xx[MEM]yy[MEM]`

2) Launch :

`[Abort]0F24[GO]`

3) Result :

`[Abort]0F20[Term]` display the GCD

Examples :

7D (125)	**5F** (95)	=>	**05** (5)
AF (175)	**32** (50)	=>	**19** (25)
FC (252)	**60** (96)	=>	**0C** (12)

```
:100F2000000000000C0FB03F8F9C8F8981F061C1C5D
:100F30001C1C1C1C1C980CC0E8C8E8C0E5C8E2C01A
:100F4000E2C8DFC0DD03F8D9C8D890D83F00000060
:00000001FF
```

```
0f24           GCD:
0f24 c0 fb       LD Nbr1
0f26 03          SCL
0f27 f8 f9       CAD Nbr2
0f29 c8 f8       ST Nbr3
0f2b 98 1f       JZ FIN
0f2d 06          CSA
0f2e 1c          SR
0f2f 1c          SR
0f30 1c          SR
0f31 1c          SR
0f32 1c          SR
0f33 1c          SR
0f34 1c          SR
0f35 98 0c       JZ CAL
0f37 c0 e8       LD Nbr1
0f39 c8 e8       ST Nbr3
0f3b c0 e5       LD Nbr2
0f3d c8 e2       ST Nbr1
0f3f c0 e2       LD Nbr3
0f41 c8 df       ST Nbr2
0f43           CAL:
0f43 c0 dd       LD Nbr2
0f45 03          SCL
0f46 f8 d9       CAD Nbr1
0f48 c8 d8       ST Nbr2
0f4a 90 d8       JMP GCD
0f4c           FIN:
0f4c 3F          XPPC 3
```

Mk14 Micro Computer

program

TITLE / TITRE: 2-byte addition (Version 2)

2-byte addition with carry.
Relative addressing of data in P2: P2 = 0F00
input : **0F** => **0FFB** P2H **00** => **0FFC** P2L

*Application of the example from the manual “**MK14 MicroComputer Training Manual**” page 45*

1) Numbers entry :
go to address 0F20
then input first number xxxx in two parts
[Abort]0F20[Term]xx[MEM]xx[MEM]
then input second number yyyy in two parts
yy[MEM]yy[MEM]

2) Launch :
[Abort]0F24[GO]

3) Result :
[Abort]0F22[Term] display the first byte of the result
[MEM] display the second byte of the result

```
:0A0F2000000000002C223F221CA03
:090F2A0023C222F220CA223F007A
:020FFB000F00E5
:00000001FF
```

```
0f24            .OR      0xf24
       0020     Nbr1H    .EQ 0x020
       0021     Nbr1L    .EQ 0x021
       0022     Nbr2H    .EQ 0x022
       0023     Nbr2L    .EQ 0x023
0f24            ADD:
0f24 02         CCL
0f25 c2 23      LD Nbr2L(2)
0f27 f2 21      ADD Nbr1L(2)
0f29 ca 23      ST Nbr2L(2)
0f2b c2 22      LD Nbr2H(2)
0f2d f2 20      ADD Nbr1H(2)
0f2f ca 22      ST Nbr2H(2)
0f31 3f         XPPC 3
```

2-byte addition with carry.

*Application of the example from the manual "**MK14 MicroComputer Training Manual**" page 45*

1) Numbers entry :
   go to address 0F20
   then input first number xxxx in two parts
   [Abort]0F20[Term]xx[MEM]xx[MEM]
   then input second number yyyy in two parts
   yy[MEM]yy[MEM]

2) Launch :
   [Abort]0F24[GO]

3) Result :

[Abort]0F22[Term]	display the first byte of the result
[MEM]	display the second byte of the result

```
:0A0F2000000000002C0FDF0F9C857
:090F2A00F9C0F6F0F2C8F23F0034
:00000001FF
```

```
0f24            .OR  0xf24
     0f20       Nbr1H .EQ 0xf20
     0f21       Nbr1L .EQ 0xf21
     0f22       Nbr2H .EQ 0xf22
     0f23       Nbr2L .EQ 0xf23


0f24            ADD:
0f24 02         CCL
0f25 c0 fd      LD Nbr2L
0f27 f0 f9      ADD Nbr1L
0f29 c8 f9      ST Nbr2L
0f2b c0 f6      LD Nbr2H
0f2d f0 f2      ADD Nbr1H
0f2f c8 f2      ST Nbr2H
0f31 3f         XPPC 3
```

Mk14 Micro Computer

program

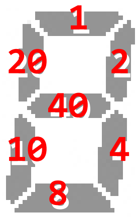
TITLE / TITRE Display text

Displaying 8-character text

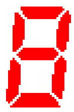
7-segment displays
==================

The MK14 features eight 7-segment displays, located in the upper part.

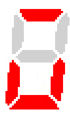
For each 7-segment display, a single byte value can define which segments are displayed.
Each illuminated segment corresponds to a hexadecimal value.
If a segment is not illuminated, its value is 0.



The sum of the hexadecimal values of all segments determines the displayed character.



1+2+4+8+10+20+40 = 7F



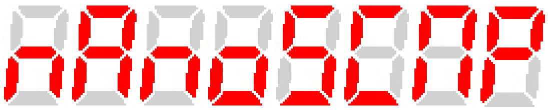
1+0+4+8+10+0+0 = 1D

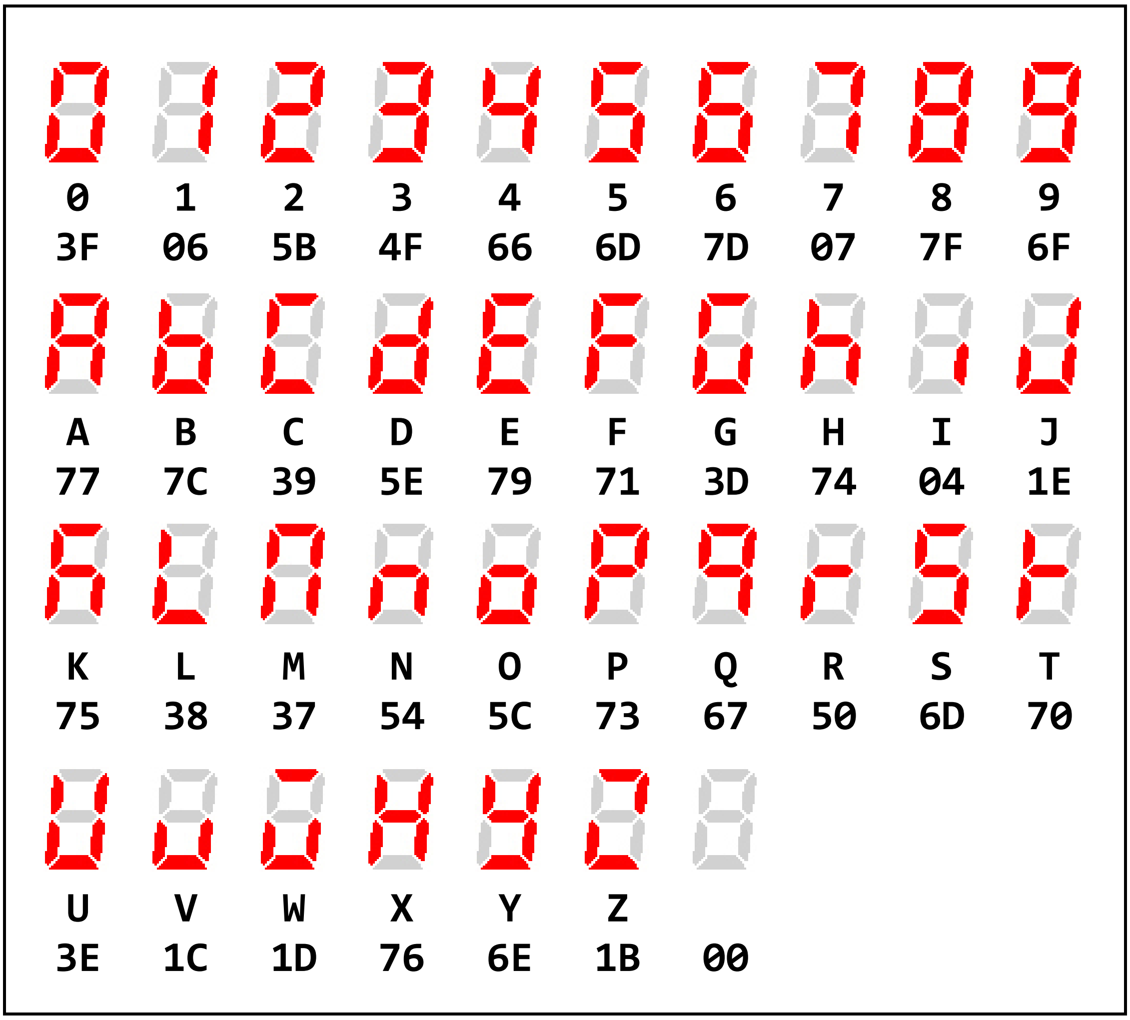
```
1) Initialize pointer P2 to 0F20
   Input : 0F => 0FFB (P2H)
           20 => 0FFC (P2L)

2) Text of 8 characters, starting from the end, from address 0F20 to 0F27 :
          0F20 73    P
          0F21 37    M
          0F22 39    C
          0F23 6D    S
          0F24 5C    O
          0F25 54    N
          0F26 77    A
          0F27 54    N

3) Display :
   go to address 185 then GO
            [Abort]185[GO]

   gives :
```





Mk14 Micro Computer

program

TITLE / TITRE: Moving message

PAGE 1 OF 1

*Application of the example from the manual “**MK14 MicroComputer Training Manual**” page 100*

Original

YOU-CAN-SEE-S-OF-C-IF-A-JOLLY-FINE-BUY-

```
:180F2000C40D35C40031C40F36C47332C460C8F0C40701C280C9808F8A
:180F380001C4FF027094F3B8DF9CEDC6FF94E590DD8079796D4037774C
:180F500039403E3F6E406D77406E3E7F4079373071406E38383F1F4049
:130F680077406D30403940713F406D4079796D4037773946
:00000001FF
```

nano-SCMP

```
0F20: C4 0D 35 C4 31 C4 0F 36 C4 73 32 C4 60 C8 F0 C4
0F30: 07 01 C2 80 C9 80 8F 01 C4 FF 02 70 94 F3 B8 DF
0F40: 9C ED C6 FF 94 E5 90 DD 80 73 37 39 6D 40 5C 54
0F50: 77 54 40 73 37 39 6D 40 5C 54 77 54 40 73 37 39
0F60: 6D 40 5C 54 77 54 40 73 37 39 6D 40 5C 54 77 54
0F70: 40 73 37 39 6D 40 5C 54 77 54 40 00 00 00 00 00
```

```
:100F2000C40D35C431C40F36C47332C460C8F0C4B4
:100F30000701C280C9808F01C4FF027094F3B8DF3B
:100F40009CEDC6FF94E590DD807337396D405C54AD
:100F5000775440733739 6D405C5477544073373958
:100F60006D405C547754407337396D405C5477540E
:100F7000407337396D405C547754400000000000E6
:00000001FF
```

CLONES-PHWEB-ME

```
0F20: C4 0D 35 C4 31 C4 0F 36 C4 73 32 C4 60 C8 F0 C4
0F30: 07 01 C2 80 C9 80 8F 01 C4 FF 02 70 94 F3 B8 DF
0F40: 9C ED C6 FF 94 E5 90 DD 80 79 37 40 7c 79 1d 74
0F50: 73 40 6d 79 54 5c 38 39 00 79 37 40 7c 79 1d 74
0F60: 73 40 6d 79 54 5c 38 39 00 79 37 40 7c 79 1d 74
0F70: 73 40 6d 79 54 5c 38 39 00 00 00 00 00 00 00 00
```

```
:100F2000C40D35C431C40F36C47332C460C8F0C4B4
:100F30000701C280C9808F01C4FF027094F3B8DF3B
:100F40009CEDC6FF94E590DD807937407c791d7477
:100F500073406d79545c3839007937407c791d7461
:100F600073406d79545c3839007937407c791d7451
:100F700073406d79545c383900000000000000000B7
:00000001FF
```