previous:
The Ciel MSX2+ Turbo

MCCW issue 93, June-December 2000
Back to contents

next:
Kyokugen

the next level in hi-speed programming

Assembly line

Assembly programmers sometimes have problems with calculations. Some of them are quite simple as there are instructions for them, others might be a bit more complicated. Adding and subtracting values are common issue with the Z80. Multiplication and division are not. The R800 has an advantage when it comes to multiplication as it has an instruction for it. And introducing the next step in MSX Assembly resources... The MSX Assembly Page.

Albert Beevendorp

Directory

References

1. DIVIWORD.ASC
2. MULUBYTE.ASC

Mathematics in another way

Many articles have been written about the math-pack which is incorporated in the main-rom of any MSX computer. I found some routines which can be made easily and a lot faster in assembly. The routine IADD (i.e. HL=DE+HL) can be done with just one assembly instruction: ADD HL,DE but to be conform to what IADD does an EX DE,HL has to be added. The routine ISUB (i.e. HL=DE-HL) can be done in very much the same way, with the only exception the register exchange is necessary. The routines IMULT (i.e. HL=DE*HL, the routine published here isn't the fastest multiplication routine though) and IDIV (i.e. HL=DE\HL) are included in this little source as well and note that IMOD (i.e. HL=DE MOD HL) uses the same routine as IDIV, with the exception the modulus is returned in HL:

ML-listing: MATHPACK.ASM

```
IADD:   EX    DE,HL           ; HL = DE + HL
        ADD   HL,DE
        RET

ISUB:   EX    DE,HL           ; HL = DE - HL
        AND   A               ; Reset Carry
        SBC   HL,DE
        RET

IMULT:  EX    DE,HL           ; HL = DE * HL
        PUSH  HL              ; Keep for subtraction later
        PUSH  HL              ; Counter
        POP   BC
IMLTLP: ADD   HL,DE           ; ADD DE to HL BC times
        DEC   BC
        LD    A,B             ; Check if counter reached zero
        OR    C
        JR    NZ,IMLTLP
        POP   DE              ; Result is DE too high
        SBC   HL,DE           ; so correct this
        RET

IDIV:   EX    DE,HL           ; HL = DE \ HL
        LD    A,E             ; Division by zero?
        OR    D
        JR    NZ,IDIVDO       ; Continue of not
        LD    HL,0
        LD    DE,0
        SCF                   ; Set Carry to indicate an error
        RET
IDIVDO: LD    C,L             ; AC = HL
        LD    A,H
        LD    HL,0            ; HL = Modulus
        LD    B,16            ; 16-bit division
        OR    A               ; Start with carry reset
IDIVLP: RL    C               ; AC = AC * 2
        RLA                   ; (set Carry to bit 0)
        RL    L               ; HL = HL * 2
        RL    H               ; (reset Carry again (HL = 0))
        PUSH  HL
        SBC   HL,DE
```

3. MULUWORD.ASC
4. DIVIBYTE.ASC

```
        CCF                     ; Carry set when Ok
        JR    C,IDIVDN          ; Modulus cannot be >= Value
        EX    (SP),HL           ; Restore Modulus
IDIVDN: INC   SP                ; Get Modulus from Stack
        INC   SP
        DJNZ  IDIVLP            ; Repeat for all bits
; Shift last Carry-bit to divider
        EX    DE,HL             ; DE = Modulus
        RL    C                 ; Carry to C
        LD    L,C               ; HL = AC
        RLA
        LD    H,A
        OR    A                 ; Reset Carry (valid result)
        RET

IMOD:   CALL  IDIV              ; HL = DE MOD HL
        EX    DE,HL             ; Modulus in HL
        RET
```

Explanations
The source contains 5 major calls which result in 4 routines. The first two are addition and subtraction. Because of the simplicity of these routines explaining how these work would be nonsense now. The routines for multiplication, division and modulus mathematics are explained right now.

Multiplication using `IMULT`
There are a few methods for multiplying one value with another. In a way, two methods are explained in this first Assembly Line. The first method is quite simple to make and understand. I agree on the fact there are methods which are much faster and a second and faster solution will be provided in the next Assembly Line. In here I have taken a rather lazy approach. The routine actually does `DE * HL - DE`. That is because I didn't reset one of the register pairs first. We need to multiply 123 with 234. The routine does the following, a step by step explanation: We make 123 a counter which is usually put in `BC`. The loop does `HL = HL + DE` exactly 123 times, so `HL = 123 + 234`, so `HL = 357`. Then `HL = 357 + 234`, so `HL = 581`. Then `HL = 581 + 234`, so `HL = 815` and after 123 times the result in `HL` would be 28905 because we started with 123 already in `HL`. So to correct the result we need to subtract 123 from `HL`. `HL` will become 28782.

Division using `IDIV`
Just like multiplication there are a few methods to divide a value from another and in a way, two methods are explained in this first Assembly Line. First, we check whether we are going to attempt an illegal division by zero. According to correct math only 1 division by zero is legal, but not tested in this routine.

Modulus using `IMOD`
For `IMOD` the exact same routine is called like `IDIV` and by swapping `DE` and `HL` the modulus result is written in `HL`. For more information about the actual division routine, read "Division using `IDIV`".

The source of `IDIV` and `IMOD` can be found right with the following reference [1] as the original file doesn't have these routines.

More math appeared with the MSX turbo R, which contains two very useful instructions to do very fast multiplication: `MULUB` and `MULUW`. Note that both instructions are located in the R800 processor and the Z80 processor still needs routines to do the multiplication part. `MULUB` needs two 8-bit registers as input and the 16-bit result is output in `HL`.

MULUW is a bit different. It needs two 16-bit registers as input and the 32-bit result is output to registers DE and HL (format DEHL). Clicking on the instruction the Z80 equivalents of MULUB A,B (reference [2]) and MULUW HL,BC (reference [3]) can be found. Both routines check whether they are running on Z80 or R800 CPU, so both the R800 instruction as the Z80 routine which does the same multiplication are included in both sources including the check. I am aware of the fact a little bug remains in the Z80 equivalent of MULUW HL,BC.

Another way of dividing

Referring to a telephone conversation I once had with Jan van der Meer — who has passed away a few years ago — of a way to divide a number by 6 the easy way, he came up with a different solution to divide two bytes by each other. The source can be found in this reference [4] and in a nutshell, this is how it works: the routine subtracts the divider from the value you want the value from until you reach 0 or a value which lies below the divider. This will be the modulus. The result is the number of times the divider could go into the value.

MSX Assembly Page

A lot has happened between the release of the last two issues of MCCW. A dedicated website for MSX assembly programmers (either current or future ones) has become a quite worthy place with resources, articles, sources, downloads and links to other pages which have info about MSX assembly quite useful to programmers. Together with the MSX Resource Center development forum the MSX Assembly Page will be the most valuable to all MSX assembly programmers. There is also an MSX development IRC channel. Information and the channel location, as well as a CGI:IRC client to it is available at MSX Banzai!.

I hope to meet many developers on the MSX development IRC channel. I also think we can help each other with a variety of programming problems and solutions, faster routines and other MSX development related issues.

A short note from the editors:

This article isn't finished at all. We think that it lacks a thorough explanation of what is going on. The author did not have time to improve it, though. It may be beneficial to very experienced MSX programmers anyway. Of course I do not need to mention that there will be no Assembly Line part 2 and beyond.

previous:
The Ciel MSX2+ Turbo

MSX Computer & Club Webmagazine
issue 93, June-December 2000

next:
Kyokugen