asm_sample.lst

GAS LISTING asm_sample.s page 1

```
   1                    #
   2                    # asm_sample.s : divide subroutine
   3                    #       2003 November    MIZUNO Y.
   4                    #
   5                    # calling sequence
   6                    #   void  divide( long long dividend, int divisor, int *quotient, int *remainder );
   7                    #
   8
   9                    /*** define constant(s) ***/
  10
  11                    .equ    INT_MIN,   0x80000000    # minimum negative value (= -2147483648)
  12
  13
  14                    /*** unsigned divide routine       ***/
  15                    /***    edx:eax / ebx -> eax & edx ***/
  16                    /***    if overflow then set CF    ***/
  17                            .text
  18                            .align  4
  19                    ui_divide:
  20 0000 55                    pushl   %ebp            # save ebp
  21 0001 89E5                  movl    %esp, %ebp      #  & make a new frame
  22 0003 53                    pushl   %ebx            # save registers
  23 0004 56                    pushl   %esi
  24 0005 57                    pushl   %edi
  25
  26                            /** check overflow **/
  27 0006 39DA                  cmpl    %ebx, %edx
  28 0008 7331                  jae     ui_div_ovf      # quotient takes more than 32 bit
  29
  30                            /** division loop **/
  31 000a B9000000              movl    $0, %ecx        # ebx:ecx = a divisor << 32
  31      00
  32 000f BF000000              movl    $0x80000000,%edi # edi: bit pattern to set quotient
  32      80
  33 0014 BE000000              movl    $0, %esi        # esi: quotient
  33      00
  34                    10:
  35 0019 D1EB                  shrl    $1, %ebx        # shift a divisor 1 bit right
  36 001b D1D9                  rcrl    $1, %ecx
  37 001d 39DA                  cmpl    %ebx, %edx      # divisor < dividend ?
  38 001f 720C                  jb      30f
  39 0021 7704                  ja      20f
  40 0023 39C8                  cmpl    %ecx, %eax      # compare lower word
  41 0025 7206                  jb      30f
  42                    20:                             # Here, dividend (edx:eax) >= shifted divisor (ebx:ecx)
  43 0027 29C8                  subl    %ecx, %eax
  44 0029 19DA                  sbbl    %ebx, %edx      # subtract divisor (ebx:ecx) from dividend (edx:eax)
  45 002b 09FE                  orl     %edi, %esi      # set 1 bit of a quotient
  46                    30:
  47 002d D1EF                  shrl    $1, %edi        # to the next bit
  48 002f 75E8                  jnz     10b             # repeat from bit-31 to bit-0
  49
  50                            /** end **/
  51 0031 89C2                  movl    %eax, %edx      # edx <- remainder
  52 0033 89F0                  movl    %esi, %eax      # eax <- quotient
  53 0035 F8                    clc                     # normal return
  54                    ui_div_ret:
```

行番号

相対アドレス

生成された機械語

アセンブリ言語

/*と*/の間、および#以下は、コメント

```
 55 0036 5F             popl    %edi            # restore registers
 56 0037 5E             popl    %esi
 57 0038 5B             popl    %ebx
 58 0039 5D             popl    %ebp
 59 003a C3             ret
 60
 61             ui_div_ovf:
 62 003b F9             stc                     # error return
 63 003c EBF8           jmp     ui_div_ret
 64
 65             /*************************/
 66             /***    main routine    ***/
 67             /*************************/
 68 003e 6690           .align  4
 69                     .globl  _divide
 70             _divide:
 71 0040 55             pushl   %ebp            # save ebp
 72 0041 89E5           movl    %esp, %ebp      #  & make a new frame
 73 0043 53             pushl   %ebx            # save registers
 74 0044 56             pushl   %esi
 75
 76                     /** check signs of a dividend and a divisor **/
 77 0045 BE000000       movl    $0, %esi        # esi: sign flag
 77      00
 78 004a 8B4508         movl    8(%ebp),  %eax  # low word of a dividend
 79 004d 8B550C         movl    12(%ebp), %edx  # high word  〃     〃
 80 0050 83FA00         cmpl    $0, %edx
 81 0053 790F           jns     10f
 82 0055 BE030000       movl    $3, %esi        # dividend is negative
 82      00
 83 005a F7D0           notl    %eax            # change a dividend edx:eax to positive
 84 005c F7D2           notl    %edx
 85 005e 83C001         addl    $1, %eax        # (make 1's complement and add 1)
 86 0061 83D200         adcl    $0, %edx
 87             10:
 88 0064 8B5D10         movl    16(%ebp), %ebx  # get a divisor
 89 0067 83FB00         cmpl    $0, %ebx        # is negative ?
 90 006a 7905           jns     20f
 91 006c 83F602         xorl    $2, %esi        # bit 2: dividend & divisor have different signs
 92 006f F7DB           negl    %ebx            # change a divisor to positive
 93
 94                     /** divide as unsigned integers **/
 95             20:
 96 0071 E88AFFFF       call    ui_divide       # edx:eax / ebx
 96      FF
 97 0076 722E           jc      div_overflow    # overflow
 98
 99                     /** negate results according to signs **/
100 0078 F7C60100       testl   $1, %esi        # dividend is negative ?
100      0000
101 007e 7402           jz      30f
102 0080 F7DA           negl    %edx            # negate a remainder
103            30:
104 0082 F7C60200       testl   $2, %esi        # dividend & divisor have different signs ?
104      0000
105 0088 7409           jz      40f
106 008a F7D8           negl    %eax            # negate a quotient
```

```
·GAS LISTING asm_sample.s           page 3


 107 008c 83F800            cmpl    $0, %eax
 108 008f 7F15              jg      div_overflow    # if not negative value then overflow
 109 0091 EB05              jmp     div_ret
 110                 40:
 111 0093 83F800            cmpl    $0, %eax
 112 0096 780E              js      div_overflow    # if negative value then overflow
 113
 114                        /** store the results **/
 115                 div_ret:
 116 0098 8B5D14            movl    20(%ebp), %ebx
 117 009b 8903              movl    %eax, (%ebx)    # store the quotient
 118 009d 8B5D18            movl    24(%ebp), %ebx
 119 00a0 8913              movl    %edx, (%ebx)    # store the remainder
 120
 121                        /** end **/
 122 00a2 5E                popl    %esi            # restore registers
 123 00a3 5B                popl    %ebx
 124 00a4 5D                popl    %ebp
 125 00a5 C3                ret
 126
 127                        /** overflow **/
 128                 div_overflow:
 129 00a6 B8000000          movl    $INT_MIN, %eax  # quotient  <- INT_MIN
 129      80
 130 00ab BA000000          movl    $INT_MIN, %edx  # remainder <- INT_MIN
 130      80
 131 00b0 EBE69090          jmp     div_ret
 131      90909090
 131      90909090
 131      90909090
·GAS LISTING asm_sample.s           page 4


DEFINED SYMBOLS
                           *ABS*:00000000 fake
        asm_sample.s:19    .text:00000000 ui_divide
        asm_sample.s:61    .text:0000003b ui_div_ovf
        asm_sample.s:54    .text:00000036 ui_div_ret
        asm_sample.s:70    .text:00000040 _divide
        asm_sample.s:128   .text:000000a6 div_overflow
        asm_sample.s:115   .text:00000098 div_ret

NO UNDEFINED SYMBOLS
```