3/19/2008

.lss file generated for a C program consisting of an empty main() function.

D:\My Documents\MSOE\Courses\Example Programs\CE2810\GNUGCC\default\GNUGCC.lss

```
GNUGCC.elf:     file format elf32-avr

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000000a6  00000000  00000000  00000054  2**1
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .stab         00000378  00000000  00000000  000000fc  2**2
                  CONTENTS, READONLY, DEBUGGING
  2 .stabstr      0000005f  00000000  00000000  00000474  2**0
                  CONTENTS, READONLY, DEBUGGING
  3 .debug_aranges 00000020  00000000  00000000  000004d3  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_pubnames 0000001b  00000000  00000000  000004f3  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_info   00000123  00000000  00000000  0000050e  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_abbrev 00000034  00000000  00000000  00000631  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   00000058  00000000  00000000  00000665  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_frame  00000020  00000000  00000000  000006c0  2**2
                  CONTENTS, READONLY, DEBUGGING
Disassembly of section .text:

00000000 <__vectors>:
   0:   0c 94 2a 00     jmp 0x54        ; 0x54 <__ctors_end>
   4:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
   8:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
   c:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  10:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  14:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  18:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  1c:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  20:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  24:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  28:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  2c:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  30:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  34:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  38:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  3c:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  40:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  44:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  48:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  4c:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>
  50:   0c 94 47 00     jmp 0x8e        ; 0x8e <__bad_interrupt>

00000054 <__ctors_end>:
  54:   11 24           eor r1, r1      ; clear r1
  56:   1f be           out 0x3f, r1    ; 63   clear SREG
  58:   cf e3           ldi r28, 0x3F   ; 63
  5a:   d8 e0           ldi r29, 0x08   ; 8    initialize Stack Pointer (SPL, SPH)
  5c:   de bf           out 0x3e, r29   ; 62
  5e:   cd bf           out 0x3d, r28   ; 61

00000060 <__do_copy_data>:
  60:   10 e0           ldi r17, 0x00   ; 0
  62:   a0 e6           ldi r26, 0x60   ; 96   load addr of SRAM start to X
  64:   b0 e0           ldi r27, 0x00   ; 0
  66:   e6 ea           ldi r30, 0xA6   ; 166  load addr of Flash data start to Z
  68:   f0 e0           ldi r31, 0x00   ; 0
  6a:   02 c0           rjmp    .+4             ; 0x70 <__do_copy_data_start>

0000006c <__do_copy_data_loop>:
```

size of program (see back)

Information regarding how each section of memory is used

initialization of all interrupt vectors

jump ahead to 0x70 (next page)

```
  6c:   05 90           lpm r0, Z+
  6e:   0d 92           st  X+, r0

00000070 <.do copy data start>:
  70:   a0 36           cpi r26, 0x60   ; 96
  72:   b1 07           cpc r27, r17
  74:   d9 f7           brne    .-10        ; 0x6c <.do_copy_data_loop>

00000076 <  do clear bss>:
  76:   10 e0           ldi r17, 0x00   ; 0
  78:   a0 e6           ldi r26, 0x60   ; 96
  7a:   b0 e0           ldi r27, 0x00   ; 0
  7c:   01 c0           rjmp    .+2         ; 0x80 <.do_clear_bss_start>

0000007e <.do clear bss loop>:
  7e:   1d 92           st  X+, r1

00000080 <.do clear bss start>:
  80:   a0 36           cpi r26, 0x60   ; 96
  82:   b1 07           cpc r27, r17
  84:   e1 f7           brne    .-8         ; 0x7e <.do_clear_bss_loop>
  86:   0e 94 49 00     call    0x92    ; 0x92 <main>
  8a:   0c 94 52 00     jmp 0xa4    ; 0xa4 <_exit>

0000008e <  bad interrupt>:
  8e:   0c 94 00 00     jmp 0   ; 0x0 <__vectors>

00000092 <main>:
  92:   cf 93           push    r28
  94:   df 93           push    r29
  96:   cd b7           in  r28, 0x3d   ; 61
  98:   de b7           in  r29, 0x3e   ; 62
  9a:   80 e0           ldi r24, 0x00   ; 0
  9c:   90 e0           ldi r25, 0x00   ; 0
  9e:   df 91           pop r29
  a0:   cf 91           pop r28
  a2:   08 95           ret

000000a4 < exit>:
  a4:   ff cf           rjmp    .-2         ; 0xa4 <_exit>
```

copy from Flash to SRAM

test for end of data

clear r17 overflow counter

load addr of SRAM start to X

clear data memory

check for end of data

call main function

prepare to make subroutine calls

loop forever