```
////////////////////////////////////////////////////////////////////////
//      var.c program to test memory allocation in C
//         M. Mizuno (c) 1995, 2004, 2005
//          modified for Learning Tree course 223P
//
// to compile var.c on Pentium,
// 1. execute Visual Studio .NET2003 (~2017) x86 Native Tools Command Prompt
// 2. go to the directory which contains var.c
// 3. issue cl /Od /FAcs var.c
//        /Od: disable optimization
//        /FAcs: generate a listing file with source code and machine code
////////////////////////////////////////////////////////////////////////
long test(unsigned int ui, int i, short s, unsigned short us,
        char c, unsigned char uc, long l, unsigned long ul,
        int x, short y);

char ret;
int x=100;
static int si;
static int sj = 23;

int main(int argc, char **argv, char **envp)
{
        unsigned char uc;
        static short y = 99;
        short s;
        char c;
        unsigned short us;
        static int i;
        unsigned int ui;
        long l;
        unsigned long ul;

        if (i < 0) {
                ui = us + s - c;
        }
        else {
                ul = si - sj * 2;
        }

        while (sj > 0) {
                uc = y - 3;
                sj++;
        }

        ret = test(ui, i, s, us, c, uc, l, ul, x,  y);

        return 0;
}

long test(unsigned int ui, int i, short s, unsigned short us,
        char c, unsigned char uc, long l, unsigned long ul,
        int x, short y)
{
        char c1;
        int i1;
        char c2;

        ui = 1;
        i = 2;
        s = 3;
        us = 4;
        c = 5;
        uc = 6;
        l = 7;
        ul = 8;
        x = 9;
        y = 10;
        c1 = 11;
        c2 = 12;
        i1 = 13;
        return ui * 2 + l;
}
```


74

```
; Listing generated by Microsoft (R) Optimizing Compiler Version 14.00.50727.42

        TITLE   C:\LearningTree\223P\Programs\memory_test\Pentium\var.c
        .686P
        .XMM
        include listing.inc
        .model  flat

INCLUDELIB LIBCMT
INCLUDELIB OLDNAMES

PUBLIC  _x
_DATA   SEGMENT
COMM    _ret:BYTE
_x      DD      064H
_sj     DD      017H
?y@?1??main@@9@9 DW 063H                                ; 'main'::'2'::y
_DATA   ENDS
PUBLIC  _test
PUBLIC  _main
_BSS    SEGMENT
_si     DD      01H DUP (?)
?i@?1??main@@9@9 DD 01H DUP (?)                         ; 'main'::'2'::i
; Function compile flags: /Odtp
; File c:\learningtree\223p\programs\memory_test\pentium\var.c
_BSS    ENDS
_TEXT   SEGMENT
_c$ = -25                                               ; size = 1
_us$ = -24                                              ; size = 2
_l$ = -20                                               ; size = 4
_ul$ = -16                                              ; size = 4
_uc$ = -9                                               ; size = 1
_ui$ = -8                                               ; size = 4
_s$ = -4                                                ; size = 2
_argc$ = 8                                              ; size = 4
_argv$ = 12                                             ; size = 4
_envp$ = 16                                             ; size = 4
_main   PROC

; 23   : {

  00000 55              push    ebp
  00001 8b ec           mov     ebp, esp
  00003 83 ec 1c        sub     esp, 28                           ; 0000001cH

; 24   :        unsigned char uc;
; 25   :        static short y = 99;
; 26   :        short s;
; 27   :        char c;
; 28   :        unsigned short us;
; 29   :        static int i;
; 30   :        unsigned int ui;
; 31   :        long l;
; 32   :        unsigned long ul;
; 33   :
; 34   :        if (i < 0) {

  00006 83 3d 00 00 00
        00 00           cmp     DWORD PTR ?i@?1??main@@9@9, 0
  0000d 7d 15           jge     SHORT $LN4@main

; 35   :                ui = us + s - c;

  0000f 0f b7 45 e8     movzx   eax, WORD PTR _us$[ebp]
  00013 0f bf 4d fc     movsx   ecx, WORD PTR _s$[ebp]
  00017 03 c1           add     eax, ecx
  00019 0f be 55 e7     movsx   edx, BYTE PTR _c$[ebp]
  0001d 2b c2           sub     eax, edx
  0001f 89 45 f8        mov     DWORD PTR _ui$[ebp], eax

; 36   :        }
; 37   :        else {
```



```
  00022 eb 12           jmp     SHORT $LN2@main
$LN4@main:

; 38   :                ul = si - sj * 2;

  00024 a1 00 00 00 00  mov     eax, DWORD PTR _sj
  00029 d1 e0           shl     eax, 1
  0002b 8b 0d 00 00 00
        00              mov     ecx, DWORD PTR _si
  00031 2b c8           sub     ecx, eax
  00033 89 4d f0        mov     DWORD PTR _ul$[ebp], ecx
$LN2@main:

; 39   :        }
; 40   :
; 41   :        while (sj > 0) {

  00036 83 3d 00 00 00
        00 00           cmp     DWORD PTR _sj, 0
  0003d 7e 1c           jle     SHORT $LN1@main

; 42   :                uc = y - 3;

  0003f 0f bf 15 00 00
        00 00           movsx   edx, WORD PTR ?y@?1??main@@9@9
  00046 83 ea 03        sub     edx, 3
  00049 88 55 f7        mov     BYTE PTR _uc$[ebp], dl

; 43   :                sj++;

  0004c a1 00 00 00 00  mov     eax, DWORD PTR _sj
  00051 83 c0 01        add     eax, 1
  00054 a3 00 00 00 00  mov     DWORD PTR _sj, eax

; 44   :        }

  00059 eb db           jmp     SHORT $LN2@main
$LN1@main:

; 45   :
; 46   :        ret = test(ui, i, s, us, c, uc, l, ul, x,  y);

  0005b 0f bf 0d 00 00
        00 00           movsx   ecx, WORD PTR ?y@?1??main@@9@9
  00062 51              push    ecx
  00063 8b 15 00 00 00
        00              mov     edx, DWORD PTR _x
  00069 52              push    edx
  0006a 8b 45 f0        mov     eax, DWORD PTR _ul$[ebp]
  0006d 50              push    eax
  0006e 8b 4d ec        mov     ecx, DWORD PTR _l$[ebp]
  00071 51              push    ecx
  00072 0f b6 55 f7     movzx   edx, BYTE PTR _uc$[ebp]
  00076 52              push    edx
  00077 0f be 45 e7     movsx   eax, BYTE PTR _c$[ebp]
  0007b 50              push    eax
  0007c 0f b7 4d e8     movzx   ecx, WORD PTR _us$[ebp]
  00080 51              push    ecx
  00081 0f bf 55 fc     movsx   edx, WORD PTR _s$[ebp]
  00085 52              push    edx
  00086 a1 00 00 00 00  mov     eax, DWORD PTR ?i@?1??main@@9@9
  0008b 50              push    eax
  0008c 8b 4d f8        mov     ecx, DWORD PTR _ui$[ebp]
  0008f 51              push    ecx
  00090 e8 00 00 00 00  call    _test
  00095 83 c4 28        add     esp, 40                           ; 00000028H
  00098 a2 00 00 00 00  mov     BYTE PTR _ret, al

; 47   :
; 48   :        return 0;

  0009d 33 c0           xor     eax, eax
```

```
; 49   : }

  0009f 8b e5            mov     esp, ebp
  000a1 5d               pop     ebp
  000a2 c3               ret     0
_main   ENDP
; Function compile flags: /Odtp
_c2$ = -6                                               ; size = 1
_c1$ = -5                                               ; size = 1
_i1$ = -4                                               ; size = 4
_ui$ = 8                                                ; size = 4
_i$ = 12                                                ; size = 4
_s$ = 16                                                ; size = 2
_us$ = 20                                               ; size = 2
_c$ = 24                                                ; size = 1
_uc$ = 28                                               ; size = 1
_l$ = 32                                                ; size = 4
_ul$ = 36                                               ; size = 4
_x$ = 40                                                ; size = 4
_y$ = 44                                                ; size = 2
_test   PROC

; 54   : {

  000b0 55               push    ebp
  000b1 8b ec            mov     ebp, esp
  000b3 83 ec 08         sub     esp, 8

; 55   :        char c1;
; 56   :        int i1;
; 57   :        char c2;
; 58   :
; 59   :        ui = 1;

  000b6 c7 45 08 01 00
        00 00            mov     DWORD PTR _ui$[ebp], 1

; 60   :        i = 2;

  000bd c7 45 0c 02 00
        00 00            mov     DWORD PTR _i$[ebp], 2

; 61   :        s = 3;

  000c4 66 c7 45 10 03
        00               mov     WORD PTR _s$[ebp], 3

; 62   :        us = 4;

  000ca 66 c7 45 14 04
        00               mov     WORD PTR _us$[ebp], 4

; 63   :        c = 5;

  000d0 c6 45 18 05      mov     BYTE PTR _c$[ebp], 5

; 64   :        uc = 6;

  000d4 c6 45 1c 06      mov     BYTE PTR _uc$[ebp], 6

; 65   :        l = 7;

  000d8 c7 45 20 07 00
        00 00            mov     DWORD PTR _l$[ebp], 7

; 66   :        ul = 8;

  000df c7 45 24 08 00
        00 00            mov     DWORD PTR _ul$[ebp], 8

; 67   :        x = 9;

  000e6 c7 45 28 09 00
```



```
        00 00            mov     DWORD PTR _x$[ebp], 9

; 68   :        y = 10;

  000ed 66 c7 45 2c 0a
        00               mov     WORD PTR _y$[ebp], 10  ; 0000000aH

; 69   :        c1 = 11;

  000f3 c6 45 fb 0b      mov     BYTE PTR _c1$[ebp], 11 ; 0000000bH

; 70   :        c2 = 12;

  000f7 c6 45 fa 0c      mov     BYTE PTR _c2$[ebp], 12 ; 0000000cH

; 71   :        i1 = 13;

  000fb c7 45 fc 0d 00
        00 00            mov     DWORD PTR _i1$[ebp], 13        ; 0000000dH

; 72   :        return ui * 2 + l;

  00102 8b 45 08         mov     eax, DWORD PTR _ui$[ebp]
  00105 8b 4d 20         mov     ecx, DWORD PTR _l$[ebp]
  00108 8d 04 41         lea     eax, DWORD PTR [ecx+eax*2]

; 73   : }

  0010b 8b e5            mov     esp, ebp
  0010d 5d               pop     ebp
  0010e c3               ret     0
_test   ENDP
_TEXT   ENDS
END
```