# 16.317: Microprocessor Systems Design I

Spring 2013

Lecture 19: Key Questions
March 20, 2013

1. Describe the issues involved in accessing data in assembly, including the two general factors the compiler must account for.

2. How does a program handle statically allocated data (data allocated at compile time)?

3. How does a program handle data that are dynamically allocated when a function is called?

4. Describe the structure of a typical x86 stack frame.

5. Describe how array accesses are handled.

6. Describe how conditional statements are handled.

7. Describe how loops are handled.

```
#include <stdio.h>

void main() {

    int X[10], Y[10];   // integer arrays
    int i, j;           // index variables

    for (i = 0; i < 10; i++) {      // outer loop
        X[i] = i * 2;           // set X[i]
        for (j = 0; j < 10; j++) {  // inner loop
            if (j < 5)          // set Y[j]
                Y[j] = X[i] + j;        //  based on
            else                //  value of j
                Y[j] = X[i] - j;
        }
    }
}
```

```
; Listing generated by Microsoft (R) Optimizing Compiler Version 16.00.40219.01

    TITLE   C:\Users\Michael_Geiger\Documents\courses\16.317_micros_I\f12\misc\hll_assembly_test\
    hll_assembly_test\testfile.c
    .686P
    .XMM
    include listing.inc
    .model  flat

INCLUDELIB MSVCRTD
INCLUDELIB OLDNAMES

PUBLIC  __$ArrayPad$
PUBLIC  _main
EXTRN   ___security_cookie:DWORD
EXTRN   @__security_check_cookie@4:PROC
EXTRN   @_RTC_CheckStackVars@8:PROC
EXTRN   __RTC_Shutdown:PROC
EXTRN   __RTC_InitBase:PROC
;   COMDAT rtc$TMZ
; File c:\users\michael_geiger\documents\courses\16.317_micros_i\f12\misc\hll_assembly_test\
    hll_assembly_test\testfile.c
rtc$TMZ SEGMENT
__RTC_Shutdown.rtc$TMZ DD FLAT:__RTC_Shutdown
rtc$TMZ ENDS
;   COMDAT rtc$IMZ
rtc$IMZ SEGMENT
__RTC_InitBase.rtc$IMZ DD FLAT:__RTC_InitBase
; Function compile flags: /Odtp /RTCsu /ZI
rtc$IMZ ENDS
;   COMDAT _main
_TEXT   SEGMENT
_j$ = -120                      ; size = 4
_i$ = -108                      ; size = 4
_Y$ = -96                       ; size = 40
_X$ = -48                       ; size = 40
__$ArrayPad$ = -4                   ; size = 4
_main   PROC                        ; COMDAT

; 3    : void main() {

    push    ebp
    mov ebp, esp
    sub esp, 316                ; 0000013cH
    push    ebx
    push    esi
    push    edi
    lea edi, DWORD PTR [ebp-316]
    mov ecx, 79                 ; 0000004fH
    mov eax, -858993460             ; ccccccccH
    rep stosd
    mov eax, DWORD PTR ___security_cookie
    xor eax, ebp
    mov DWORD PTR __$ArrayPad$[ebp], eax

; 4    :
; 5    :    int X[10], Y[10];   // integer arrays
; 6    :    int i, j;           // index variables
; 7    :
; 8    :    for (i = 0; i < 10; i++) {      // outer loop

    mov DWORD PTR _i$[ebp], 0
    jmp SHORT $LN8@main
$LN7@main:
    mov eax, DWORD PTR _i$[ebp]
    add eax, 1
```

```
    mov DWORD PTR _i$[ebp], eax
$LN8@main:
    cmp DWORD PTR _i$[ebp], 10          ; 0000000aH
    jge SHORT $LN9@main

; 9    :        X[i] = i * 2;           // set X[i]

    mov eax, DWORD PTR _i$[ebp]
    shl eax, 1
    mov ecx, DWORD PTR _i$[ebp]
    mov DWORD PTR _X$[ebp+ecx*4], eax

; 10   :        for (j = 0; j < 10; j++) {  // inner loop

    mov DWORD PTR _j$[ebp], 0
    jmp SHORT $LN5@main
$LN4@main:
    mov eax, DWORD PTR _j$[ebp]
    add eax, 1
    mov DWORD PTR _j$[ebp], eax
$LN5@main:
    cmp DWORD PTR _j$[ebp], 10          ; 0000000aH
    jge SHORT $LN3@main

; 11   :            if (j < 5)          // set Y[j]

    cmp DWORD PTR _j$[ebp], 5
    jge SHORT $LN2@main

; 12   :                Y[j] = X[i] + j;        //  based on

    mov eax, DWORD PTR _i$[ebp]
    mov ecx, DWORD PTR _X$[ebp+eax*4]
    add ecx, DWORD PTR _j$[ebp]
    mov edx, DWORD PTR _j$[ebp]
    mov DWORD PTR _Y$[ebp+edx*4], ecx

; 13   :            else                //  value of j

    jmp SHORT $LN1@main
$LN2@main:

; 14   :                Y[j] = X[i] - j;

    mov eax, DWORD PTR _i$[ebp]
    mov ecx, DWORD PTR _X$[ebp+eax*4]
    sub ecx, DWORD PTR _j$[ebp]
    mov edx, DWORD PTR _j$[ebp]
    mov DWORD PTR _Y$[ebp+edx*4], ecx
$LN1@main:

; 15   :        }

    jmp SHORT $LN4@main
$LN3@main:

; 16   :    }

    jmp SHORT $LN7@main
$LN9@main:

; 17   : }

    xor eax, eax
    push    edx
    mov ecx, ebp
```

```
    push    eax
    lea edx, DWORD PTR $LN14@main
    call    @_RTC_CheckStackVars@8
    pop eax
    pop edx
    pop edi
    pop esi
    pop ebx
    mov ecx, DWORD PTR __$ArrayPad$[ebp]
    xor ecx, ebp
    call    @__security_check_cookie@4
    mov esp, ebp
    pop ebp
    ret 0
    npad    2
$LN14@main:
    DD  2
    DD  $LN13@main
$LN13@main:
    DD  -48                 ; ffffffd0H
    DD  40                  ; 00000028H
    DD  $LN11@main
    DD  -96                 ; ffffffa0H
    DD  40                  ; 00000028H
    DD  $LN12@main
$LN12@main:
    DB  89                  ; 00000059H
    DB  0
$LN11@main:
    DB  88                  ; 00000058H
    DB  0
_main   ENDP
_TEXT   ENDS
END
```

```c
#include <stdio.h>

int a, b, c;

void main() {
    scanf("%d %d %d", &a, &b, &c);
    printf("a = %d, b = %d, c = %d\n", a, b, c);
}
```

```
; Listing generated by Microsoft (R) Optimizing Compiler Version 16.00.40219.01

    TITLE   C:\Users\Michael_Geiger\Documents\courses\16.317_micros_I\f12\misc\hll_assembly_test\
    hll_assembly_test\testfile2.c
    .686P
    .XMM
    include listing.inc
    .model  flat

INCLUDELIB MSVCRTD
INCLUDELIB OLDNAMES

_DATA   SEGMENT
COMM    _a:DWORD
COMM    _c:DWORD
COMM    _b:DWORD
_DATA   ENDS
PUBLIC  ??_C@_0BI@HLEICADJ@a?5?$DN?5?$CFd?0?5b?5?$DN?5?$CFd?0?5c?5?$DN?5?$CFd?6?$AA@ ; `string'
PUBLIC  ??_C@_08OOHKHLPO@?$CFd?5?$CFd?5?$CFd?$AA@   ; `string'
PUBLIC  _main
EXTRN   __imp__printf:PROC
EXTRN   __imp__scanf:PROC
EXTRN   __RTC_CheckEsp:PROC
EXTRN   __RTC_Shutdown:PROC
EXTRN   __RTC_InitBase:PROC
;   COMDAT ??_C@_0BI@HLEICADJ@a?5?$DN?5?$CFd?0?5b?5?$DN?5?$CFd?0?5c?5?$DN?5?$CFd?6?$AA@
; File c:\users\michael_geiger\documents\courses\16.317_micros_i\f12\misc\hll_assembly_test\
    hll_assembly_test\testfile2.c
CONST   SEGMENT
??_C@_0BI@HLEICADJ@a?5?$DN?5?$CFd?0?5b?5?$DN?5?$CFd?0?5c?5?$DN?5?$CFd?6?$AA@ DB 'a'
    DB  ' = %d, b = %d, c = %d', 0aH, 00H       ; `string'
CONST   ENDS
;   COMDAT ??_C@_08OOHKHLPO@?$CFd?5?$CFd?5?$CFd?$AA@
CONST   SEGMENT
??_C@_08OOHKHLPO@?$CFd?5?$CFd?5?$CFd?$AA@ DB '%d %d %d', 00H ; `string'
CONST   ENDS
;   COMDAT rtc$TMZ
rtc$TMZ SEGMENT
__RTC_Shutdown.rtc$TMZ DD FLAT:__RTC_Shutdown
rtc$TMZ ENDS
;   COMDAT rtc$IMZ
rtc$IMZ SEGMENT
__RTC_InitBase.rtc$IMZ DD FLAT:__RTC_InitBase
; Function compile flags: /Odtp /RTCsu /ZI
rtc$IMZ ENDS
;   COMDAT _main
_TEXT   SEGMENT
_main   PROC                        ; COMDAT

; 5    : void main() {

    push    ebp
    mov ebp, esp
    sub esp, 192                ; 000000c0H
    push    ebx
    push    esi
    push    edi
    lea edi, DWORD PTR [ebp-192]
    mov ecx, 48                 ; 00000030H
    mov eax, -858993460             ; ccccccccH
    rep stosd

; 6    :    scanf("%d %d %d", &a, &b, &c);

    mov esi, esp
    push    OFFSET _c
```

```
    push    OFFSET _b
    push    OFFSET _a
    push    OFFSET ??_C@_08OOHKHLPO@?$CFd?5?$CFd?5?$CFd?$AA@
    call    DWORD PTR __imp__scanf
    add esp, 16                 ; 00000010H
    cmp esi, esp
    call    __RTC_CheckEsp

; 7    :    printf("a = %d, b = %d, c = %d\n", a, b, c);

    mov esi, esp
    mov eax, DWORD PTR _c
    push    eax
    mov ecx, DWORD PTR _b
    push    ecx
    mov edx, DWORD PTR _a
    push    edx
    push    OFFSET ??_C@_0BI@HLEICADJ@a?5?$DN?5?$CFd?0?5b?5?$DN?5?$CFd?0?5c?5?$DN?5?$CFd?6?$AA@
    call    DWORD PTR __imp__printf
    add esp, 16                 ; 00000010H
    cmp esi, esp
    call    __RTC_CheckEsp

; 8    : }

    xor eax, eax
    pop edi
    pop esi
    pop ebx
    add esp, 192                ; 000000c0H
    cmp ebp, esp
    call    __RTC_CheckEsp
    mov esp, ebp
    pop ebp
    ret 0
_main   ENDP
_TEXT   ENDS
END
```