Chapter 2_2

C Pointer Arithmetic and Correspoinding Assembly Code

Assembly Code for Arrays and Pointers

- To analyze assembly code generated for arrays and pointers, we use the following notation:
- Assume the following declarations:
  - int i; // i: integer variable i
  - int ia[3][4][5]; // ia: int array [3][4][5]
  - int (*ip)[4][5] = ia; // ip : pointer-to-int variable ip

1. Address in bss/data
   - _ip : address of ip (variable in bss/data)
   - _i : address of i (variable in bss/data)
   - _ia : starting adderss of array ia (constant in bss/data)
2. Address in a stack frame
   1. _ip$: offset of _ip (relative to fp)
      - _ip$[fp] is the address of ip
   2. _i$: offset of _i
   3. _ia$: offset of _ia

Assembly Code for Arrays and Pointers

3. Contents
   - @<_ip> : contents of address _ip in bss/data
   - @<_i$[fp]> : contents of address _i$[fp] (when i is in a stack frame)
   - @<_ia$[fp]> : ia[0][0][0] // contents of [0][0][0] (the 1st element) of array ia in a stack frame

Example

- Again, consider the following external variable declarations:

```
int  ia[3][4][5];

int  (*ip)[4][5] = ia;
```

- Then, expression ip[2][3][4] is analyzed as follows:

1. ip
   - T: ptr to [4][5] of int
   - V: @<_ip>
2. ip + 2
   - T: ptr to [4][5] of int
   - V: @<_ip> + 2 * sizeof ([4][5] of int) = @<_ip> + 2 * 4 * 5 * 4 = @<_ip> + 160
3. *(ip + 2)
   - T: [4][5] of int = ptr to [5] of int
   - V: @<_ip> + 160

Example (cont)

4. *(ip + 2) + 3
   - T: ptr to [5] of int
   - V: @<_ip> + 160 + 3 * sizeof([5] of int) = @<_ip> + 160 + 60 = @<_ip> + 220
5. *(*(ip + 2) + 3)
   - T: [5] of int = ptr to int
   - V: @<_ip> + 220
6. *(*(ip + 2) + 3) + 4
   - T: ptr to int
   - V: @<_ip> + 220 + 4 * sizeof(int) = @<_ip> + 236
7. *(*(*(ip + 2) + 3) + 4)
   - T: int
   - V: @<@<_ip> + 236>

Example (cont)

```
; 10   :  x = ip[2][3][4];

0000e    b8 50 00 00 00          mov      eax, 80                    ; 00000050H
00013    d1 e0                   shl      eax, 1
00015    03 05 00 00 00
         00                      add      eax, DWORD PTR _ip
0001b    b9 14 00 00 00          mov      ecx, 20                    ; 00000014H
00020    6b d1 03      imul      edx, ecx, 3
00023    03 c2                   add      eax, edx
00025    b9 04 00 00 00          mov      ecx, 4
0002a    c1 e1 02      shl       ecx, 2
0002d    8b 14 08      mov       edx, DWORD PTR [eax+ecx]
00030    89 55 fc      mov       DWORD PTR _x$[ebp], edx
```

- The above code does:

  @< @<_ip> +80*2 + 20 * 3 + 4 * 4 > = @< @<_ip> + 236 >

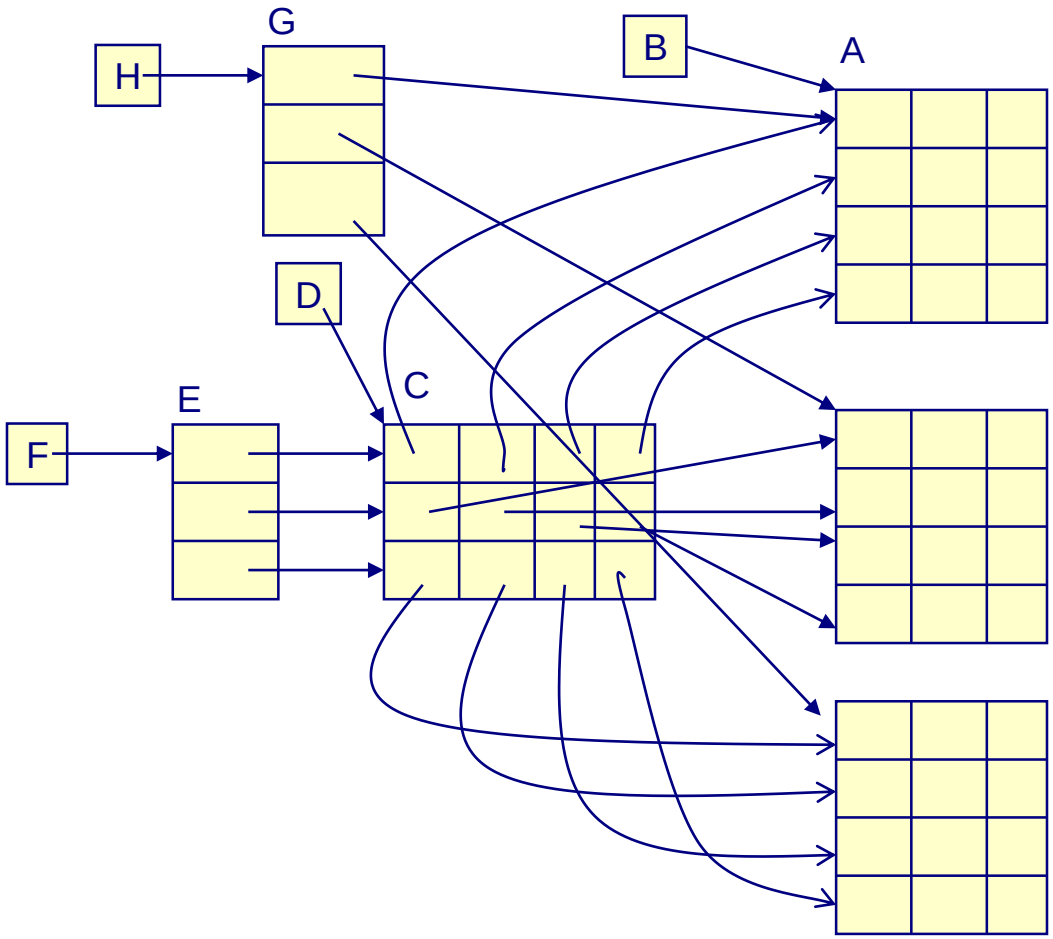
  = @< @<_ip> + sizeof([4][5] of int) * 2 + sizeof([5] of int) * 3 + sizeof(int) * 4 >

- With optimization (/O2), the code generated is:

```
00000  a1 ec 00 00 00     mov       eax, DWORD PTR _ia+236
```


More Examples

- Consider Pointer Exercise 1



More Examples (cont)

- c[i][j][k] is analyzed as follows:
  - c: [3][4] of ptr to int, and c is in “data area” (the address is _c)
  - c[i][j][k] = *(*(*(c + i) + j) + k), where i, j, and k are local variables

1. c:
   - T: [3][4] of ptr to int = ptr to [4] of ptr to int
   - V: _c
2. c + i
   - T: ptr to [4] of ptr to int
   - V: _c + i * sizeof([4] of ptr to int) = _c+ 16* @<_i$[fp]>
3. *(c + i)
   - T: [4] of ptr to int = ptr to ptr to int
   - V: _c+ 16* @<_i$[fp]>
4. *(c+i)+j
   - T: ptr to ptr to int
   - V: _c+ 16* @<_i$[fp]> + j * sizeof(ptr to int) = _c + 16* @<_i$[fp]> + 4*@<_j$[fp]>

More Examples (cont)

5. *(*(c + i) + j)
   - T: ptr to int
   - V: @<_c + 16* @<_i$[fp]> + 4*@<_j$[fp]>>

6. *(*(c + i) + j)+k
   - T: ptr to int
   - V: @<_c + 16* @<_i$[fp]> + 4*@<_j$[fp]>> + k * sizeof(int) =
     @< _c + 16* @<_i$[fp]> + 4*@<_j$[fp]> > + 4*@<_k$[fp]>

7. *(*(*(c+i)+j)+k)
   - T: int
   - V: @< @< _c + 16* @<_i$[fp]> + 4*@<_j$[fp]> > + 4*@<_k$[fp]> >

More Examples (cont)

- @< @< _c + 16* @<_i$[fp]> + 4*@<_j$[fp]> > + 4*@<_k$[fp]> >
- The code generated for this expression is:

```
; 40   :   x = c[i][j][k];

00076      8b 45 e4      mov          eax, DWORD PTR _i$[ebp]
                         ; eax = @<_i[fp]>
00079      c1 e0 04      shl          eax, 4
                          ; eax = 16*@<_i[fp]>
0007c      8b 4d ec      mov          ecx, DWORD PTR _j$[ebp]
                          ; ecx = @<_j$[fp]>
0007f      8b 94 88 00 00
           00 00         mov          edx, DWORD PTR _c[eax+ecx*4]
                         ; edx = @<  _c+16*<@<_i[fp]> + 4*@<_j$[fp]> >
00086      8b 45 e8      mov          eax, DWORD PTR _k$[ebp]
                        ; eax = @<_k$[fp]>
00089      8b 0c 82      mov          ecx, DWORD PTR [edx+eax*4]
                        ; ecx = @<  @<  _c+16*<@<_i[fp]> + 4*@<_j$[fp]> > + 4*@<_k[fp]>  >
0008c      89 4d e0      mov          DWORD PTR _x$[ebp], ecx
```


More Examples (cont)

- Next, consider e[i][j][k]:
  - e: [3] of ptr to ptr to int
  - e, i, j, and k are all local variables

1. e
   - T: [3] of ptr to ptr to int = ptr to ptr to ptr to int
   - V: _e$[fp]
2. e + i
   - T: ptr to ptr to ptr to int
   - V: _e$[fp] + i * sizeof(ptr to ptr to int) = _e$[fp] + 4*@<_i$[fp>
3. *(e + i)
   - T: ptr to ptr to int
   - V: @<_e$[fp] + 4*@<_i$[fp>>
4. *(e + i) + j
   - T: ptr to ptr to int
   - V: @<_e$[fp] + 4*@<_i$[fp>> + j * sizeof(ptr to int) =
     @<_e$[fp] + 4*@<_i$[fp>> + 4*@<_j$[fp]>

More Examples (cont)

5. *(*(e + i) + j)
   - T: ptr to int
   - V: @< @<_e$[fp] + 4*@<_i$[fp>> + 4*@<_j$[fp]>>

6. *(*(e + i) + j) + k
   - T: ptr to int
   - V: @< @<_e$[fp] + 4*@<_i$[fp>> + 4*@<_j$[fp]>> + k * sizeof(int)
     = @< @<_e$[fp] + 4*@<_i$[fp>> + 4*@<_j$[fp]>> + 4*@<_k$[fp]>

7. *(*(*(e + i) + j) + k)
   - T: int
   - V: @< @< @< _e$[fp] + 4*@<_i$[fp > > + 4*@<_j$[fp]> > + 4*@<_k$[fp]> >

More Examples (cont)

- @< @< @< _e$[fp] + 4*@<_i$[fp > > + 4*@<_j$[fp]> > + 4*@<_k$[fp]> >

```
; 42   :   x = e[i][j][k];

 000a7     8b 4d e4     mov         ecx, DWORD PTR _i$[ebp]
                         ;ecx = @<_i$[fp]>
 000aa     8b 54 8d f0  mov         edx, DWORD PTR _e$[ebp+ecx*4]
                        ; edx = @< _e$[fp] + 4*@<_i$[fp]>>
 000ae     8b 45 ec     mov         eax, DWORD PTR _j$[ebp]
                        ; eax  =@<_j$[fp]>
 000b1     8b 0c 82     mov         ecx, DWORD PTR [edx+eax*4]
                         ; ecx = @<  @<_e$[fp]+4*@<i$[fp]>>+4*<_j$[fp]>>
 000b4     8b 55 e8     mov         edx, DWORD PTR _k$[ebp]
                         ; edx = @<_k$[fp]>
 000b7     8b 04 91     mov         eax, DWORD PTR [ecx+edx*4]
                         ; eax = @<  @<  @<_e$[fp]+4*@<i$[fp]>>+4*<_j$[fp]>> + 4*@<_k$[fp]>>
 000ba     89 45 e0     mov         DWORD PTR _x$[ebp], eax
                         ;
```


More Examples (cont)

- Consider g[i][j][k]:
  - g: [3] of ptr to [3] of int
  - g is a global and i, j, k are local variables

1. g:
   - T: [3] of ptr to [3] of int = ptr to ptr to [3] of int
   - V: _g
2. g + i
   - T: ptr to ptr to [3] of int
   - V: _g + i * sizeof(ptr to [3] of int) = _g + 4*@<_i$[fp]>
3. *(g + i)
   - T: ptr to [3] of int
   - V: @<_g + 4*@<_i$[fp]>>
4. *(g + i) + j
   - T: ptr to [3] of int
   - V: @<_g + 4*@<_i$[fp]>> + j * sizeof([3] of int) =
     @<_g + 4*@<_i$[fp]>> + 12*@<_j$[fp]>

More Examples (cont)

5. *(*(g + i) + j)
   - T: [3] of int = ptr to int
   - V: @<_g + 4*@<_i$[fp]>> + 12*@<_j$[fp]>

6. *(*(g + i) + j) + k
   - T: ptr to int
   - V: @<_g + 4*@<_i$[fp]>> + 12*@<_j$[fp]> + k*sizeof(int) =
     @<_g + 4*@<_i$[fp]>> + 12*@<_j$[fp]> + 4*@<_k$[fp]>

7. *(*(*(g + i) + j) + k)
   - T: int
   - V: @<@<_g + 4*@<_i$[fp]>> + 12*@<_j$[fp]> + 4*@<_k$[fp]>>

More Examples (cont)

- @<@<_g + 4*@<_i$[fp]>> + 12*@<_j$[fp]> + 4*@<_k$[fp]>>

```
; 44   :   x = g[i][j][k];

000d5      6b 55 ec 0c                imul edx, DWORD PTR _j$[ebp], 12
                                      ; edx = 12*@<_j$[fp]>
000d9      8b 45 e4      mov          eax, DWORD PTR _i$[ebp]
                                      ; eax = @<_i$[fp]>
000dc      03 14 85 00 00
           00 00         add          edx, DWORD PTR _g[eax*4]
                                     ; edx = 12*@<j$[ebp]> + @<_g + 4*@<_i$[fp]>>
000e3      8b 4d e8      mov          ecx, DWORD PTR _k$[ebp]
                                     ; ecx = @<_k$[fp]>
000e6      8b 14 8a      mov          edx, DWORD PTR [edx+ecx*4]
                                     ; edx = edx = @<@<_g + 4*@<_i$[fp]>> + 12*@<_j$[hp]> + 4*@<_k$[fp]> >
000e9      89 55 e0      mov          DWORD PTR _x$[ebp], edx
```