華中科技大學

C程序中的函数调用揭秘

许 向 阳

xuxy @ hust. edu. cn

华中科技大学计算机科学与技术学院

有关函数调用的问题

華中科技大學

- 如何传递参数?
- 传递什么?

  按值传递、按地址传递、按引用传递

  不同类型的形参/实参，传递的内容有何差别?
- 传到什么地方去了?
- 如何进入函数?
- 如何从函数返回?
- 如何传递函数返回值?
- 函数中变量空间如何分配?
- 如何理解递归函数调用?

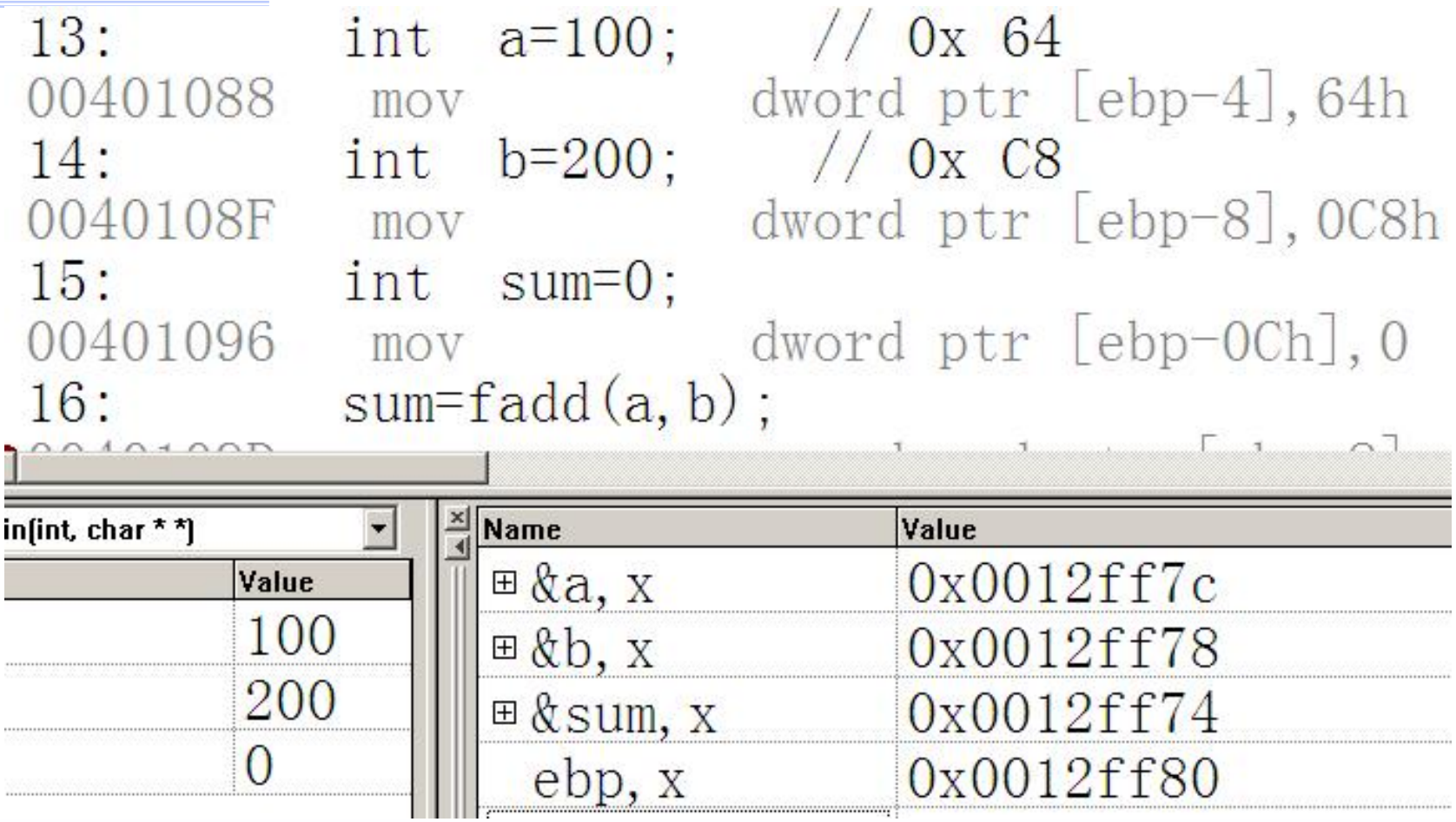
看程序实例，分析总结规律

華中科技大學

```
#include <stdio.h>
int fadd(int x, int y)
{
        int u,v,w;
        u=x+10;
        v=y+25;
        w=u+v;
        return w;
}
```

```
int main(int argc, char* argv[])
{
        int  a=100;    // 0x 64
        int  b=200;    // 0x C8
        int  sum=0;
        sum=fadd(a,b);
        printf("%d\n",sum);
        return 0;
}
```

变量空间分配



0012FF74 00 00 00 00 C8 00 00 00 64 00 00 00

函数调用

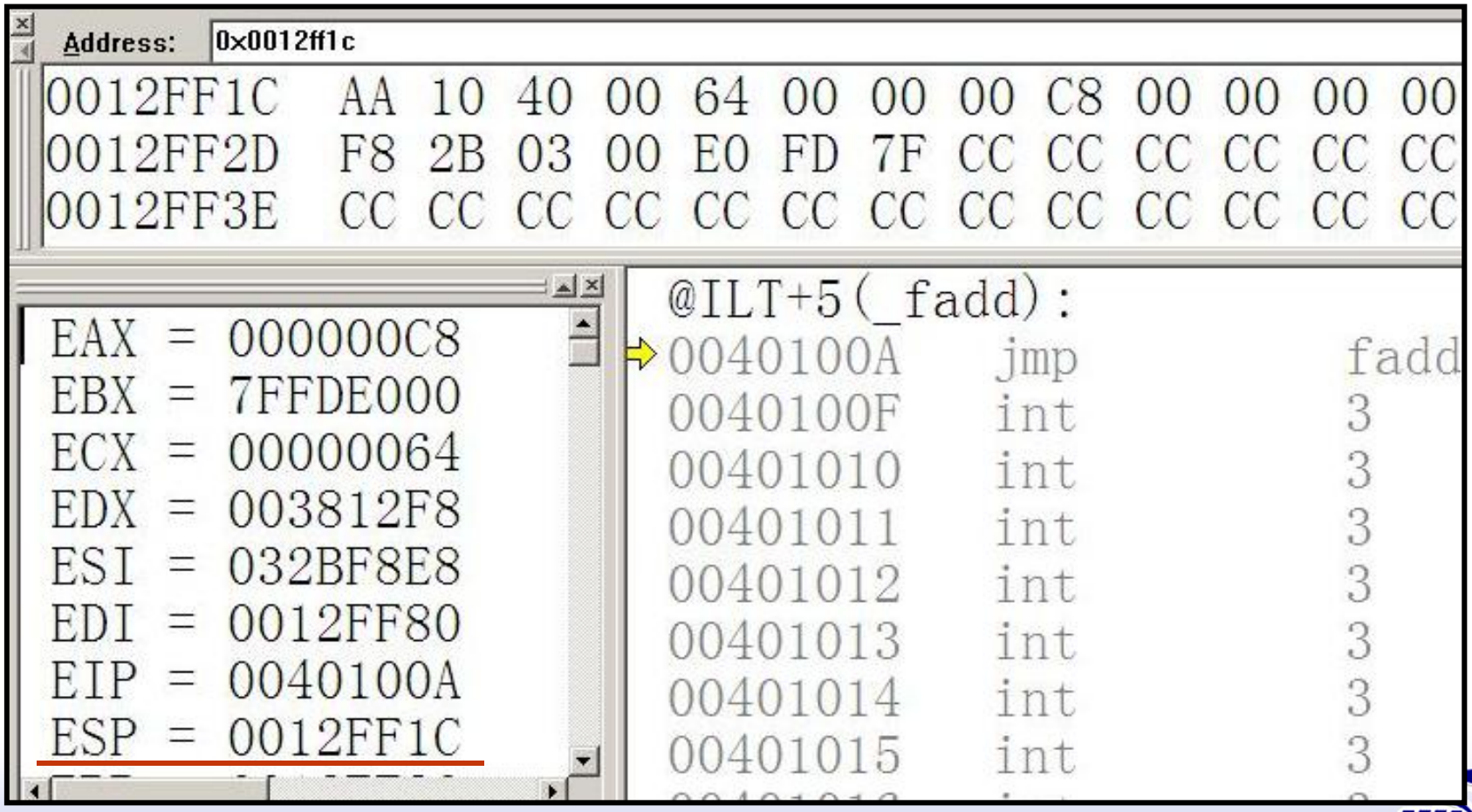
華中科技大學

```
13:         int  a=100;     // 0x 64
00401088    mov          dword ptr [ebp-4],64h
14:         int  b=200;     // 0x C8
0040108F    mov          dword ptr [ebp-8],0C8h
15:         int  sum=0;
00401096    mov          dword ptr [ebp-0Ch],0
16:         sum=fadd(a,b);
0040109D    mov          eax,dword ptr [ebp-8]
004010A0    push         eax
004010A1    mov          ecx,dword ptr [ebp-4]
004010A4    push         ecx
004010A5    call         @ILT+5(_fadd) (0040100a)
004010AA    add          esp,8
004010AD    mov          dword ptr [ebp-0Ch],eax
```

函数调用

Sum=fadd(a,b)

执行CALL指令后的状态



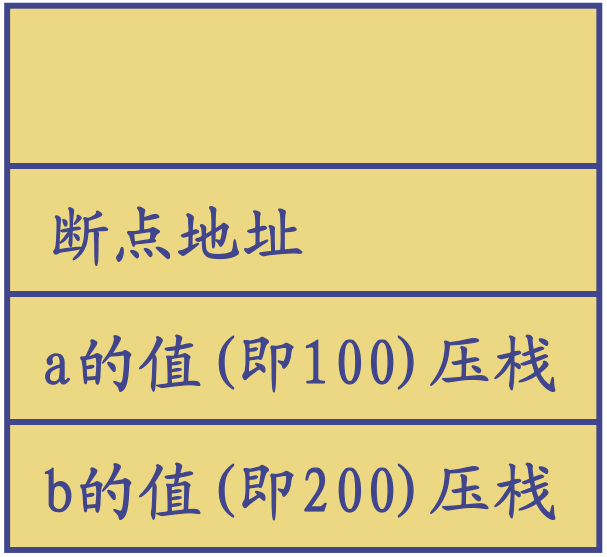
函数调用

華中科技大學

Sum=fadd(a,b)　　执行CALL指令后的状态

```
004010A5   call     @ILT+5(_fadd) (0040100a)
004010AA   add      esp,8
```

0012FF1C　ESP →



EIP 为函数的入口地址

(EIP) = 0040100A

```
0012FF1C  AA 10 40 00 64 00 00 00 C8 00 00 00
```

函数调用


```
3:    int fadd(int x, int y)
4:    {
00401020   push      ebp
00401021   mov       ebp,esp
00401023   sub       esp,4Ch
00401026   push      ebx
00401027   push      esi
00401028   push      edi
00401029   lea       edi,[ebp-4Ch]
0040102C   mov       ecx,13h
00401031   mov       eax,0CCCCCCCCh
00401036   rep stos   dword ptr [edi]
5:        int u,v,w;
6:        u=x+10;
00401038   mov       eax,dword ptr [ebp+8]
0040103B   add       eax,0Ah
0040103E   mov       dword ptr [ebp-4],eax
```

观察形参x的位置

函数调用

華中科技大學

```
5:      int u,v,w;
6:      u=x+10;
mov     eax,dword ptr [ebp+8]
add     eax,0Ah
mov     dword ptr [ebp-4],eax
7:      v=v+25;
mov     ecx,dword ptr [ebp+0Ch]
add     ecx,19h
mov     dword ptr [ebp-8],ecx
8:      w=u+v;
mov     edx,dword ptr [ebp-4]
add     edx,dword ptr [ebp-8]
mov     dword ptr [ebp-0Ch],edx
9:      return w;
mov     eax,dword ptr [ebp-0Ch]
```

观察局部变量的位置

函数调用——返回


```
9:      return w;
mov       eax,dword ptr [ebp-0Ch]
10:   }
pop       edi
pop       esi
pop       ebx
mov       esp, ebp
pop       ebp
ret
```

思考：局部变量的作用域？
局部空间的释放？
函数如何的返回？
改变形参的值，对实参有影响吗？

函数调用——返回

華中科技大學

```
15:     int  sum=0;
mov       dword ptr [ebp-0Ch],0
16:     sum=fadd(a,b);
mov       eax,dword ptr [ebp-8]
push      eax
mov       ecx,dword ptr [ebp-4]
push      ecx
call      @ILT+5(_fadd)
add       esp,8
mov       dword ptr [ebp-0Ch],eax
```

函数调用——返回

華中科技大學

```
17:         printf("%d\n",sum);
004010B0    mov         edx,dword ptr [ebp-0Ch]
004010B3    push        edx
004010B4    push        offset string "%d\n" (0042201c)
004010B9    call        printf (004010f0)
004010BE    add         esp,8
18:         return 0;
004010C1    xor         eax,eax
19:     }
004010C3    pop         edi
004010C4    pop         esi
004010C5    pop         ebx
004010C6    add         esp,4Ch
004010C9    cmp         ebp,esp
004010CB    call        __chkesp (00401170)
004010D0    mov         esp,ebp
004010D2    pop         ebp
004010D3    ret
```

函数编译——代码优化

華中科技大學

Debug版本调试中：
在局部变量之上，留了 **40H**个字节的空间？
局部变量的初始化值是多少？
保护了未用的一些的寄存器？

Release 版本

函数编译——代码优化

華中科技大學

Release 版本

```
:00401010   push 000000C8
:00401015   push 00000064
:00401017   call   00401000
:0040101C   push eax

; Possible StringData Ref from Data Obj ->"%d"
:0040101D   push 00407030
:00401022   call 00401030
:00401027   add esp, 00000010
:0040102A   xor eax, eax
:0040102C   ret

:00401000   mov eax, dword ptr [esp+08]
:00401004   mov ecx, dword ptr [esp+04]
:00401008   lea eax, dword ptr [ecx+eax+23]
:0040100C   ret
```

递归函数调用

華中科技大學

使用递归子程序 求 N!

```
#include <stdio.h>

int f(int x)
{
   if (x==1)  return 1;
   return  x*f(x-1);
}

void main()
{
  printf("%d\n",f(5));
}
```

C4_digui.c

递归函数调用

華中科技大學

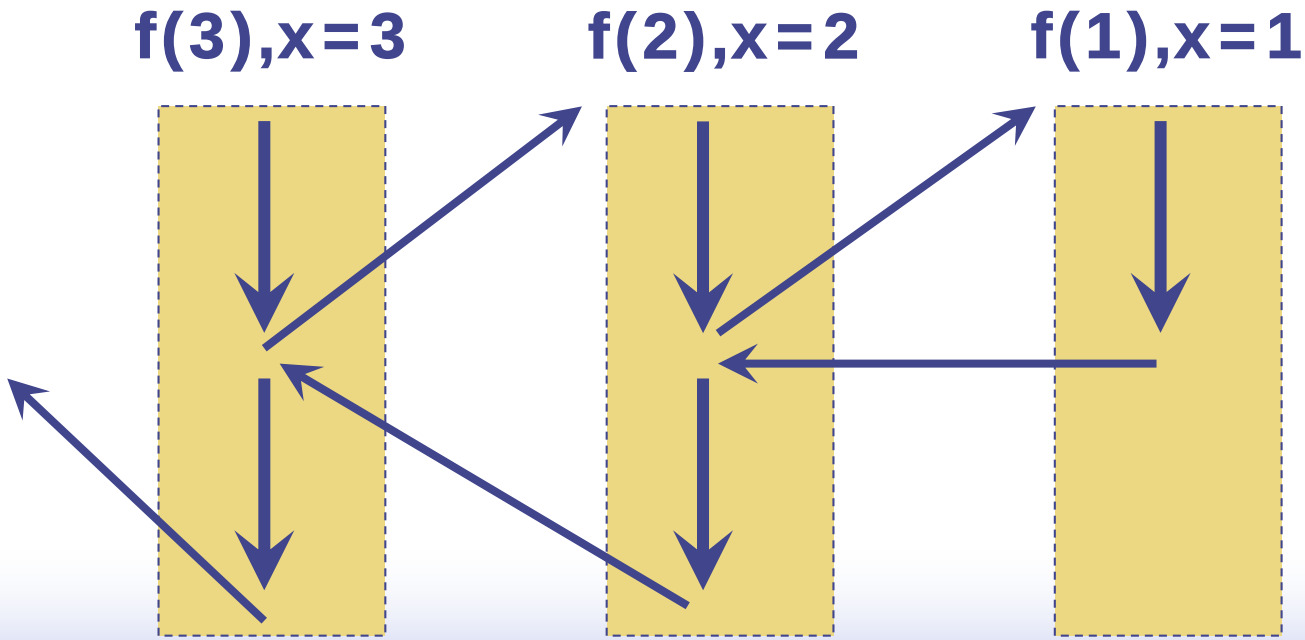
```
:00401000    push esi
:00401001    mov esi, dword ptr [esp+08]
:00401005    cmp esi, 00000001
:00401008    jne 0040100E
:0040100A    mov eax, esi
:0040100C    pop esi
:0040100D    ret

:0040100E    lea eax, dword ptr [esi-01]
:00401011    push eax
:00401012    call 00401000
:00401017    imul eax, esi
:0040101A    add esp, 00000004
:0040101D    pop esi
:0040101E    ret
```

求阶乘的子程序
Release 版本

递归函数调用

递归函数调用的理解

```
int f(int x)
{
   if (x==1)  return 1;
   return  x*f(x-1);
}
```



讨论

華中科技大學

C程序调用中，传递的入口参数，所占用的存储空间何时释放？

是在子程序中，用 **RET N** 好？
还是回到主程序后，修改 **ESP**，使之指向入口参数之下为好？

如何实现参数个数不定的函数（**printf**）？

精雕细琢——程序优化

華中科技大學

strcpy的函数实现，看汇编代码

- 一次传送一个字节吗?
- 物理上，实现一个双字（位于不同位置）的传送的速度相同吗?
  例如，从（1000H)，(1001H)分别取出一个双字送EAX。
- 如何快速判断一个双字中某个字节为 0 ？
- 用C语言，写strcpy的实现函数，可以采用哪些技巧?