计算机工作的基本过程

程序示例

```
#include <stdio.h>

int main(int argc, char* argv[])
{
        int  x, y, z;
        x = 10;
        y = 20;
        z = 3 * x + 6 * y + 4 * 8;
        printf("3*%d+6*%d+4*8=%d\n", x, y, z);
        return 0;
}
```

程序的运行

```
    int  x, y, z;
    x = 10;
00651865 C7 45 F8 0A 00 00 00 mov         dword ptr [ebp-8],0Ah
    y = 20;
0065186C C7 45 EC 14 00 00 00 mov         dword ptr [ebp-14h],14h
    z = 3 * x + 6 * y +
00651873 6B 45 F8 03                                    ,3
00651877 6B 4D EC 06                                    h],6
0065187B 8D 54 08 20
0065187F 89 55 E0                                       dx
    printf("3*%d+6*%d+4
```

寄存器

```
EAX = 0065C003 EBX = 00B6C000
ECX = 0065C003 EDX = 00000001
ESI = 00651023 EDI = 00D6FDB4
EIP = 00651865 ESP = 00D6FCC4
EBP = 00D6FDB4 EFL = 00000246
```

程序的运行

方法：程序的运行（反汇编）调试演示、讲解

- 程序执行的基本过程

  取指、译码、执行

  程序计数器（PC）（即 x86中 EIP ）

  数据的存、取

  指令执行后的变化

- 掌握反汇编窗口、寄存器窗口、监视窗口、内存窗口等

  等的用法；走进 计算机的 0、1世界

程序的运行

介绍内容：

- 汇编与反汇编
- 数据的地址及数据的表示形式

  局部变量的地址表达形式

  变量地址在监视窗口的显示

  局部变量的地址计算（EBP +n ）（验证地址表示形式）

  变量内容在内存中的显示

  监视窗口变量内容的显示（十进制，十六进制等）

程序的运行

在内存窗口观察，指令流

```
    int  x, y, z;
    x = 10;
00651865 C7 45 F8 0A 00 00 00 mov
    y = 20;
0065186C C7 45 EC 14 00 00 00 mov
    z = 3 * x + 6 * y + 4 * 8;
00651873 6B 45 F8 03          imul
00651877 6B 4D EC 06          imul
0065187B 8D 54 08 20          lea
0065187F 89 55 E0             mov
    printf("3*%d+6*%d+4*8=%d\n", x, y,
```

内存 1

地址: 0x00651865

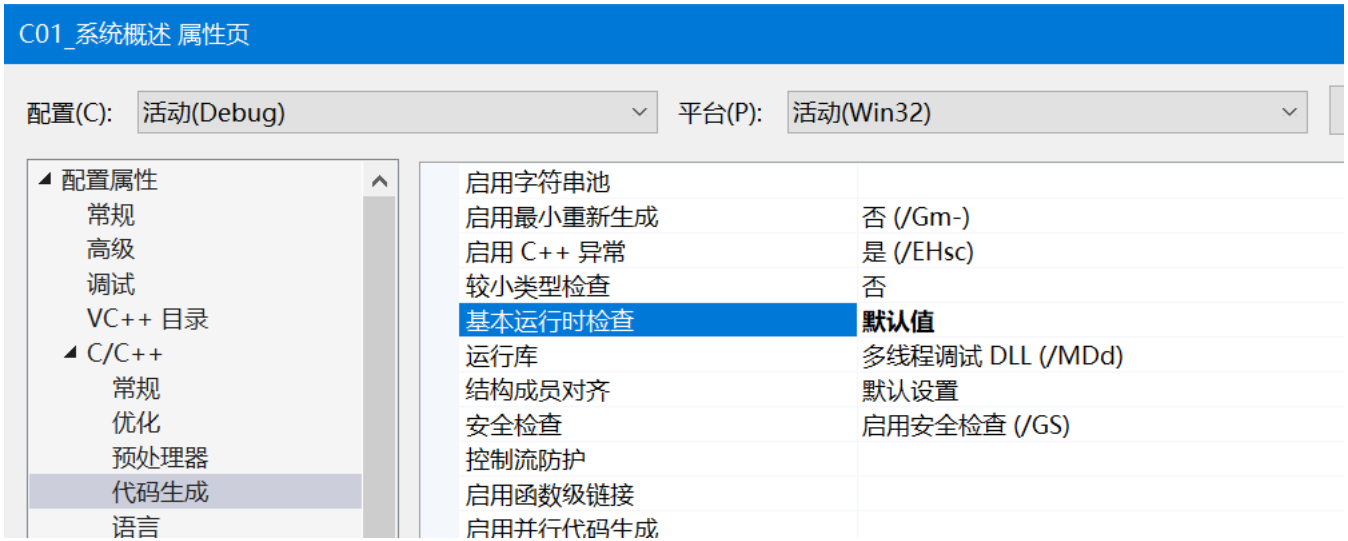
0x00651865	c7	45	f8	0a	00	00	00
0x0065186C	c7	45	ec	14	00	00	00
0x00651873	6b	45	f8	03	6b	4d	ec
0x0065187A	06	8d	54	08	20	89	55
0x00651881	e0	8b	45	e0	50	8b	4d

分析指令机器码的组成部分：帮助理解指令译码

程序的运行

分析变量之间的地址关系

- 比较不同编译开关下，编译器为变量安排空间的差异



- 比较x86, x64下，编译器为变量安排空间的差异

  理解地址分配的方案是多种多样的

程序的运行

- 在内存窗口，观察 printf 中格式串的存放
  了解参数传递的方法
- C语句与机器指令的对应关系
- 反汇编窗口，不同选项的设置，带来的窗口内容的变化
- 理解符号地址的概念