《汇编语言》作业（一）

参考答案

（1）字和双字在存储器中存放方式：字和双字在存储器中按字节被存放在连续地址中，而字与双字的长度在不同的系统中可能略有差异，Intel 80x86 系统中字等于 2 字节，即 16bit，双字等于 4 字节，即 32bit。而根据数据各字节在连续地址中排列顺序的不同，可以分为两种排列方式：大端和小端。大端方式将数据的最高有效字节存放在小地址单元中，将最低有效字节存放在大地址单元，即低地址放高位，高地址放低位。小端方式则与大端方式相反。具体示例如下图所示：

存储数据：01234567H						
地址		0800H	0801H	0802H	0803H	
大端方式	……	01H	23H	45H	67H	……
小端方式	……	67H	45H	23H	01H	……

小端方式：一种存储数据的方式。其将数据的最高有效字节存放在大地址单元中，将最低有效字节存放在小地址单元中，即低地址放低位，高地址放高位。

字和双字存储单元对齐地址：字存储单元对齐地址为 2 的倍数，双字存储单元对齐地址为 4 的倍数。

对齐地址原因：为了避免多次访存而带来指令执行效率的降低。分析：假定计算机系统中访存机制限制每次访存最多只能读取 64 位，即 8 个字节，那么第 0~7 字节可以同时读写，第 8~15 字节可以同时读写。此时如果一条指令要访问的数据不在地址为 8i~8i+7(I = 0,1,2,…)之间的存储单元内，而是交错存储，则就需要多次访存，从而延长了指令的执行时间。例如，若访问数据在第 6、7、8、9 这四个字节中，则需要访问存储器两次。

（2）逻辑地址：在 8086 内部和用户编程时，采用的“段地址:偏移地址”形式被称为逻辑地址。其中段地址说明逻辑段在主存中的起始位置，偏移地址说明主存单元距离段起始位置的偏移量，也称为虚地址。

物理地址：在 8086 中，每个存储器单元都有的一个唯一的 20 位地址，该线性排列的地址被称为该单元的物理地址，也称为实际地址。

逻辑地址转换：将该单元所在段的段寄存器中存放的 16 位段基址值左移 4 位，再加上该单元的偏移地址就得到了要访问单元的 20 位物理地址。

逻辑地址对应物理地址：

逻辑地址	物理地址
FFFF:0	FFFF0H
40:17	00417H
2000:4500	24500H
B821:4567	BC777H

（3）源操作数寻址方式：

指令	源操作数寻址方式	DX 的值	EA 的值
MOV DX, [1234H]	直接寻址	-	1234H
MOV DX, 1234H	立即寻址	1234H	-
MOV DX, BX	寄存器寻址	2000H	-
MOV DX, [BX]	寄存器间接寻址	-	2000H

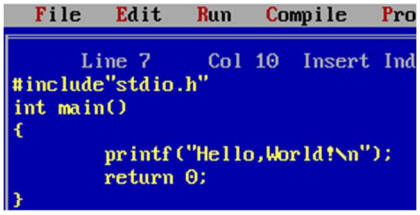
MOV DX, [BX+1234H]	相对基址寻址	-	3234H
MOV DX, [BX+DI]	基质变址寻址	-	2040H
MOV DX, [BX+DI+1234H]	相对基址变址寻址		3274H

（4）AX 寄存器内容:

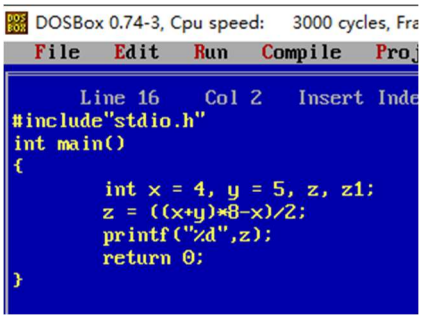
指令	指令执行后 AX 内容
MOV AX, 1200H	1200H
MOV AX, BX	0100H
MOV AX, [1200H]	4C2AH
MOV AX, [BX]	3412H
MOV AX, [BX+1100H]	4C2AH
MOV AX, [BX+SI]	7856H
MOV AX, [BX][SI+1100H]	65B7H

附加题：

代码实现：

<table>
<tr><th>例程</th><th>C 语言</th><th>汇编语言</th></tr>
<tr><td>(a)</td><td> 
</td><td>

```
DSEG SEGMENT
    ;此处输入数据段代码
    MESS DB 'Hello,World!',0DH,0AH,24H
DSEG ENDS

STACKS SEGMENT WORD STACK
    ;此处输入堆栈段代码
    DW 256 DUP(?)
STACKS ENDS

CODES SEGMENT
    ASSUME CS:CODES,DS:DSEG,SS:STACKS
START:
    MOV AX,DSEG
    MOV DS,AX

    MOV DX,OFFSET MESS
    MOV AH,9
    INT 21H
    MOV AH,4CH
    INT 21H
CODES ENDS
    END START

```

</td></tr>
<tr><td>(b)</td><td> 
</td><td>

```
; calculate.asm
; calculate z=((x+y)*8-x)/2
DATAS SEGMENT
    X DB 4
    Y DB 5
    Z DB ?
    Z1 DB ?
DATAS ENDS

CODES SEGMENT
    ASSUME CS:CODES,DS:DATAS,SS:STACKS
START:
    MOV AX,DATAS
    MOV DS,AX

    MOV AL,X
    ADD AL,Y
    MOV BL,8
    IMUL BL
    MOV BL,X
    MOV BH,0
    SUB AX,BX
    MOV BL,2
    IDIV BL
    MOV Z,AL
    MOV Z1,AH

    MOV AL,Z
    MOV AH,0
    MOV BL,10
    DIV BL
    MOV DX,AX
    ADD DX,3030H
    MOV AH,2
    INT 21H
    MOV DL,DH
    MOV AH,2
    INT 21H

    MOV AH,4CH
    INT 21H
CODES ENDS
    END START
```

</td></tr>
</table>

运行结果：

例程	C 语言	汇编语言
(a)	D:\>123.exe Hello,World!	D:\>c:\a.exe Hello,World!
(b)	D:\>123.exe 34	D:\>c:\b.exe 34

Debug 结果：

例程(a):

C 语言——

字符串存储位置——

```
-D 08A0:0000
08A0:0000  00 00 00 00 54 75 72 62-6F 2D 43 20 2D 20 43 6F   ....Turbo-C - Co
08A0:0010  70 79 72 69 67 68 74 20-28 63 29 20 31 39 38 38   pyright (c) 1988
08A0:0020  20 42 6F 72 6C 61 6E 64-20 49 6E 74 6C 2E 00 4E    Borland Intl..N
08A0:0030  75 6C 6C 20 70 6F 69 6E-74 65 72 20 61 73 73 69   ull pointer assi
08A0:0040  67 6E 6D 65 6E 74 0D 0A-44 69 76 69 64 65 20 65   gnment..Divide e
08A0:0050  72 72 6F 72 0D 0A 41 62-6E 6F 72 6D 61 6C 20 70   rror..Abnormal p
08A0:0060  72 6F 67 72 61 6D 20 74-65 72 6D 69 6E 61 74 69   rogram terminati
08A0:0070  6F 6E 0D 0A 60 10 00 F0-08 00 70 00 60 10 00 F0   on..`.....p.`...
```

```
-D
08A0:0180  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
08A0:0190  00 00 00 00 48 65 6C 6C-6F 2C 57 6F 72 6C 64 21   ....Hello,World!
08A0:01A0  0A 00 00 00 00 13 02 02-04 05 06 08 08 08 14 15   ................
08A0:01B0  05 13 FF 16 05 11 02 FF-FF FF FF FF FF FF FF FF   ................
08A0:01C0  FF FF FF FF 05 05 FF FF-FF FF FF FF FF FF FF FF   ................
08A0:01D0  FF FF FF FF FF FF 0F FF-23 02 FF 0F FF FF FF FF   ........#.......
08A0:01E0  13 FF FF 02 02 05 0F 02-FF FF FF 13 FF FF FF FF   ................
08A0:01F0  FF FF FF FF 23 FF FF FF-FF 23 FF 13 FF 00 46 02   ....#....#....F.
```

运行过程部分截图——

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG

```
AX=04BA  BX=0004  CX=0010  DX=0000  SP=FFE4  BP=FFE4  SI=003A  DI=047A
DS=08A0  ES=08A0  SS=08A0  CS=076A  IP=0522   NV UP EI PL NZ NA PO NC
076A:0522 5D            POP     BP
-T

AX=04BA  BX=0004  CX=0010  DX=0000  SP=FFE6  BP=FFF0  SI=003A  DI=047A
DS=08A0  ES=08A0  SS=08A0  CS=076A  IP=0523   NV UP EI PL NZ NA PO NC
076A:0523 C3            RET
-T

AX=04BA  BX=0004  CX=0010  DX=0000  SP=FFE8  BP=FFF0  SI=003A  DI=047A
DS=08A0  ES=08A0  SS=08A0  CS=076A  IP=038D   NV UP EI PL NZ NA PO NC
076A:038D 83C402        ADD     SP,+02
-T

AX=04BA  BX=0004  CX=0010  DX=0000  SP=FFEA  BP=FFF0  SI=003A  DI=047A
DS=08A0  ES=08A0  SS=08A0  CS=076A  IP=0390   NV UP EI NG NZ NA PO NC
076A:0390 8BD8          MOV     BX,AX
-T

AX=04BA  BX=04BA  CX=0010  DX=0000  SP=FFEA  BP=FFF0  SI=003A  DI=047A
DS=08A0  ES=08A0  SS=08A0  CS=076A  IP=0392   NV UP EI NG NZ NA PO NC
076A:0392 07            POP     ES
```

汇编语言——

字符串存储位置——

```
-D 076A:0000
076A:0000  48 65 6C 6C 6F 2C 57 6F-72 6C 64 21 0D 0A 24 00   Hello,World!..$.
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-D
076A:0080  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0090  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:00A0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

运行过程部分截图——

```
-T

AX=076A  BX=0000  CX=0220  DX=0000  SP=0200  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=076B  CS=078B  IP=0003   NV UP EI PL NZ NA PO NC
078B:0003 8ED8          MOV     DS,AX
-T

AX=076A  BX=0000  CX=0220  DX=0000  SP=0200  BP=0000  SI=0000  DI=0000
DS=076A  ES=075A  SS=076B  CS=078B  IP=0005   NV UP EI PL NZ NA PO NC
078B:0005 BA0000        MOV     DX,0000
-T

AX=076A  BX=0000  CX=0220  DX=0000  SP=0200  BP=0000  SI=0000  DI=0000
DS=076A  ES=075A  SS=076B  CS=078B  IP=0008   NV UP EI PL NZ NA PO NC
078B:0008 B409          MOV     AH,09
-T

AX=096A  BX=0000  CX=0220  DX=0000  SP=0200  BP=0000  SI=0000  DI=0000
DS=076A  ES=075A  SS=076B  CS=078B  IP=000A   NV UP EI PL NZ NA PO NC
078B:000A CD21          INT     21
```

例程(b):

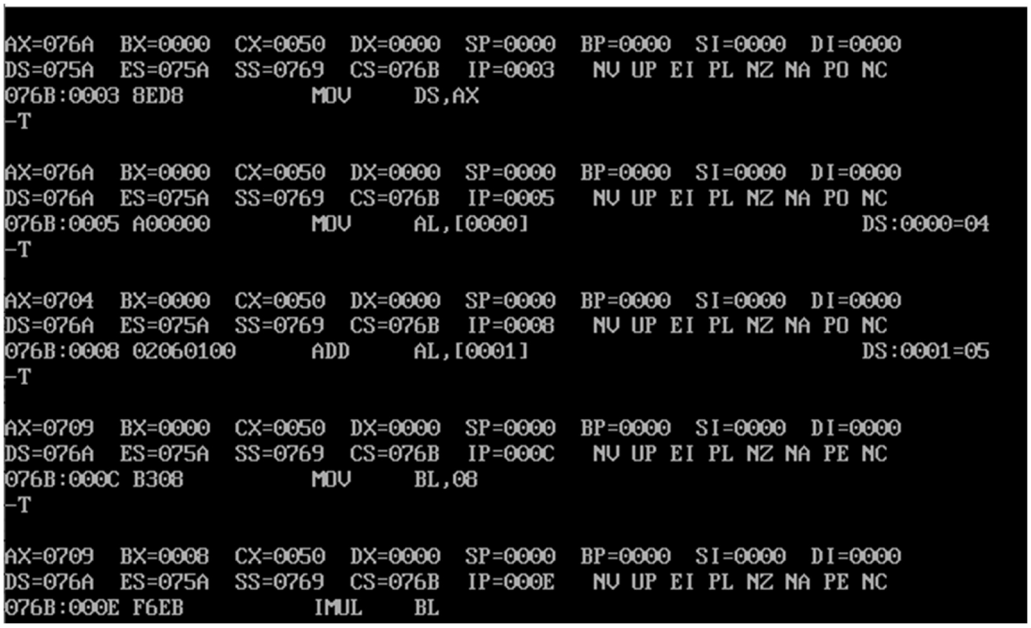
C 语言——

运行过程部分截图——

```
AX=FFFF  BX=0000  CX=2359  DX=08A3  SP=0080  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=08EA  CS=076A  IP=0003   NV UP EI PL NZ NA PO NC
076A:0003 2E            CS:
076A:0004 8916F801      MOV     [01F8],DX                       CS:01F8=0000
-T

AX=FFFF  BX=0000  CX=2359  DX=08A3  SP=0080  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=08EA  CS=076A  IP=0008   NV UP EI PL NZ NA PO NC
076A:0008 B430          MOV     AH,30
-T

AX=30FF  BX=0000  CX=2359  DX=08A3  SP=0080  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=08EA  CS=076A  IP=000A   NV UP EI PL NZ NA PO NC
076A:000A CD21          INT     21
-T

AX=30FF  BX=0000  CX=2359  DX=08A3  SP=007A  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=08EA  CS=F000  IP=14A0   NV UP DI PL NZ NA PO NC
F000:14A0 FB            STI
-T

AX=30FF  BX=0000  CX=2359  DX=08A3  SP=007A  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=08EA  CS=F000  IP=14A1   NV UP EI PL NZ NA PO NC
F000:14A1 FE38          ???     [BX+SI]                         DS:0000=CD
```

汇编语言——

运行过程部分截图——



对比与分析:

	C 语言	汇编语言
源程序	C 源程序更加简洁易懂，屏蔽了对于相关寄存器等的操作。	汇编语言代码更加长，并且可读性较差，其操作相较 C 语言更加接近底层，能够直接对寄存器等硬件内容进行修改操作。
可执行文件	C 语言可执行文件较汇编语言更大，其中包含了更多的操作。	汇编语言可执行文件相较于 C 语言更小。
结果	二者输出结果相同。	
Debug 结果	从上述 Debug 的截图可以看出，C 语言对应的汇编文件具有更多的内容，其中包括了编译器、高级语言代码等内容，除此之外，其操作更加复杂，涉及到很多子程序的调用以及数据的传递，理解起来较为困难。	相较于 C 语言的 debug 内容，汇编语言的内容更加容易理解，其执行代码即是编写的代码内容，并且相关数据的位置也较为明确。