综合性课程设计 1：MIPS 汇编器设计

要求：

用算法语言（C 或 C++）设计 MIPS 汇编与反汇编器工具，支持 MIPS 指令子集（附件）。支持不小于 10K 指令的汇编容量。最终提交必须把所有动态链接编译进去(静态连接),可独立执行程序和源代码及工程。

基本功能：

汇编：

1. 支持附录 A.10（第 3 版）或附录子集所有整数指令和伪指令。生成 2 进制目标文件.bin 和.coe 格式文件；

2. 汇编指令以";"为结束符，"//"和"#"为注释符；

3. 寄存器符号支持 R0~R31(大小写均可)和 MIPS 使用约定(第 3 版图 2-18)

反汇编：支持附录 A.10（第 3 版）所有整数指令或附录子集。支持.bin 和.coe 文件加载。

文件操作功能：能加载保存 ASM、bin 和.coe 文件

File：新建(N)　　Ctrl+N
　　打开(O)　　Ctrl+O→（3 个子项：ASM、bin 和 coe）
　　保存(S)　　Ctrl+S
　　保存为(A)　　Ctrl+Shift+S→（3 个子项：ASM、bin 和 coe）
　　打印(P)　　Ctrl+P（选做）
　　Exit(X)

以上为必做内容

扩展功能（选做）：　　**编辑：**可以编辑文本和 2、16 进制

模拟器功能

调试（Debug）：支持单步执行、单步(跳过,不进入函数)、运行到。内存 1MB，支持运行结果显示格式如下：

寄存器：IP=12345678；

R0($zero) = 00000000；　R1($at)　=12345678；　……　R7（$a3）　=12345678

R8（$t0）　=00000000；R9（$t1）=12345678；　……　R15（$t7）=12345678

R16($s0)　=00000000；　R17($s0)=12345678；　……　R23（$7）　=12345678

R24($t8)　=00000000；　R25($t9)=12345678；　……　R31($ra)=12345678

内存

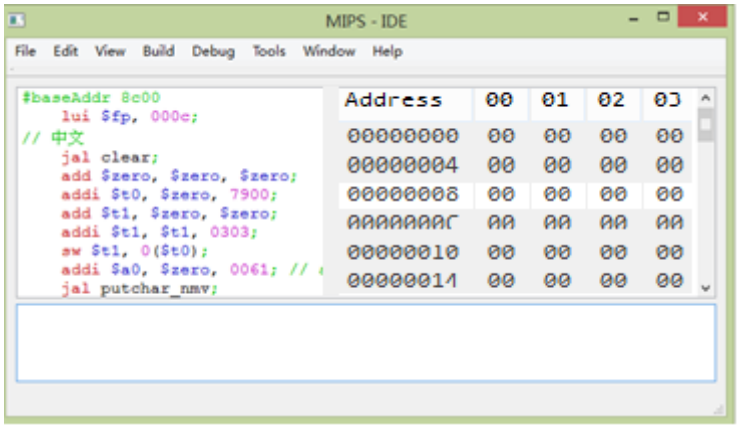
00000000：12345678 12345678 12345678 12345678 12345678 12345678 12345678 12345678

00001000：12345678 12345678 12345678 12345678 12345678 12345678 12345678 12345678

00002000：12345678 12345678 12345678 12345678 12345678 12345678 12345678 12345678

……

窗口参考格式：



菜单

File：新建(N)　　Ctrl+N
　　打开(O)　　Ctrl+O
　　保存(S)　　Ctrl+S
　　保存为(A)　　Ctrl+Shift+S
　　打印(P)　　Ctrl+P
　　Exit(X)

Edit：撤消(U)　　Ctrl+Z
　　剪切(T)　　Ctrl+X
　　复制(C)　　Ctrl+C
　　粘贴(P)　　Ctrl+V
　　删除(L)　　Del
　　查找(F)　　Ctrl+F
　　替换(R)...　　Ctrl+H
　　转到(G)　　Ctrl+G
　　全选(A)　　Ctrl+A

Bulid：汇编(Asm)
　　Coe
　　反汇编

Debug：单步(Step)
　　单步（跳过 Jal）
　　运行到…
　　停止

汇编器的一些补充说明(参考后面的例子)：

汇编器语法参考附录 A.10.2，并增加下列伪指令：

1. 指定代码起始地址：BaseAddre

```
BaseAddre: 00008c00;          //用","分隔，";"结束。可以多个定义，但后继地址
    ……;                      //必须大于前面定义的地址。中间不确定区域填"0"
```

2. 指定数据起始地址：DataAddre

DataAddre: 00008d00; //用”,”分隔，”;”结束。可以多个定义，但后继地址

DB: 0x12，0x34，……; //必须大于前面定义的地址。中间不确定区域填“0”

DW: 0x1234，0x4567，……;

二个地址申明之间的空余处填“00000000”，后继申明地址必须大于“前次申明地址+已使用空间”，否则结出错误指示。代码空间超出范围也结出错误指示

3. 初始化数据伪指令

关键字后定义指定，每个数据由逗号分隔：

DB： 定义字节数据 例： DB 0x12，0x34，……;

DW：定义 16 位字数据 例： DW 0x1234，0x4567，……;

DD： 定义 32 位字数据 DD 0x12345678，0x87654321，……;

参数定义：

EQU：定义常量 例： X EQU 12; //定义 X 为常数 12

4. 所有标号包括数据区基地址都支持伪指令：la rdest, label_address

以上为必须

声明未初始化的数据（定义数据空间）：

RESB： 定义字节空间

L1: RESB 16； //定义 16 个字节空间，L1 是标号

RESW：定义 16 位字空间

L2: RESW 16； //定义 16 个 16 位字空间，L2 是标号

RESD： 定义 32 位字节空间

L3: RESD 16 //定义 16 个 32 位字空间，L3 是标号

数据定义例子，逗号隔离(默认为 16 进制)：

```
db    0x55 ;                //ust the byte 0x55
db    0x55,0x56,0x57;       // three bytes in succession
db    'a',0x55;             //character constants are OK
db    'hello',13,10,'$' ;   // so are string constants
dw    0x1234 ;              // 0x34 0x12
dw    'a' ;                 // 0x41 0x00 (it's just a number)
dw    'ab' ;                //0x41 0x42 (character constant)
dw    'abc' ;               //0x41 0x42 0x43 0x00 (string)
dd    0x12345678 ;          //0x78 0x56 0x34 0x12
```

定义空间的例子：

```
buffer:     resb    64;   // reserve 64 bytes，buffer 是标号
wordvar:    resw    1;    // reserve a word，wordvar 是标号
```

例，代码和数据，以字地址计算：

```
#baseAddr 0000
        j       start;                  //0
        add $zero, $zero, $zero;        //4
        add $zero, $zero, $zero;        //8
        add $zero, $zero, $zero;        //C
        add $zero, $zero, $zero;        //10
        add $zero, $zero, $zero;        //14
        add $zero, $zero, $zero;        //18
        add $zero, $zero, $zero;        //1C

start:                                  //标号，可以不换行，后面直接跟代码
        nor  $at, $zero, $zero;         //r1=FFFFFFFF
        add $v1, $at, $at;              //r3=FFFFFFFE
        add $v1, $v1, $v1;              //r3=FFFFFFFC
        add $v1, $v1, $v1;              //r3=FFFFFFF8
        add $v1, $v1, $v1;              //r3=FFFFFFF0

        add $v1, $v1, $v1;              //r3=FFFFFFE0
        add $v1, $v1, $v1;              //r3=FFFFFFC0
        nor  $s4, $v1, $zero;           //r20=0000003F
        add $v1, $v1, $v1;              //r3=FFFFFF80
        add $v1, $v1, $v1;              //r3=FFFFFF00

        add $v1, $v1, $v1;              //r3=FFFFFE00
        add $v1, $v1, $v1;              //r3=FFFFFC00
        add $v1, $v1, $v1;              //r3=FFFFF800
        add $v1, $v1, $v1;              //r3=FFFFF000

        add $v1, $v1, $v1;              //r3=FFFFE000
        add $v1, $v1, $v1;              //r3=FFFFC000
        add $v1, $v1, $v1;              //r3=FFFF8000
        add $v1, $v1, $v1;              //r3=FFFF0000

        add $v1, $v1, $v1;              //r3=FFFE0000
        add $v1, $v1, $v1;              //r3=FFFC0000
        add $v1, $v1, $v1;              //r3=FFF80000
        add $v1, $v1, $v1;              //r3=FFF00000

        add $v1, $v1, $v1;              //r3=FFE00000
        add $v1, $v1, $v1;              //r3=FFC00000
```

```
        add $v1, $v1, $v1;              //r3=FF800000
        add $v1, $v1, $v1;              //r3=FF000000

        add $v1, $v1, $v1;              //r3=FE000000
        add $v1, $v1, $v1;              //r3=FC000000
        add $a2, $v1, $v1;              //r6=F8000000
        add $v1, $a2, $a2;              //r3=F0000000

        add $a0, $v1, $v1;              //r4=E0000000

        add $t5, $a0, $a0;              //r13=C0000000
        add $t0, $t5, $t5;              //r8=80000000

loop:                                   //标号，可以不换行，后面直接跟代码
        slt   $v0, $zero,$at;           //r2=00000001
        add $t6, $v0, $v0;
        add $t6, $t6, $t6;              //r14=4
        nor  $t2, $zero, $zero;         //r10=FFFFFFFF
        add $t2, $t2, $t2;              //r10=FFFFFFFE

loop1:
        sw   $a2, 4($v1);               //计数器端口:F0000004，送计数常数 r6=F8000000
        lw    $a1, 0($v1);
        add $a1, $a1, $a1;              //左移
        add $a1, $a1, $a1;
        sw   $a1, 0($v1);

        add $t1, $t1, $v0;              //r9=r9+1
        sw   $t1, 0($a0);               //r9 送 r4=E0000000 七段码端口
        lw    $t5, 14($zero);           //取存储器 20 单元预存数据至 r13,程序计数延时常数

loop2:                                  //标号，可以不换行，后面直接跟代码
        lw    $a1, 0($v1);              //读 GPIO 端口 F0000000 状态
        add $a1, $a1, $a1;
        add $a1, $a1, $a1;              //左移 2 位将 SW 与 LED 对齐，同时 D1D0 置 00，选择计数器通道 0
        sw   $a1, 0($v1);               //r5 输出到 GPIO 端口 F0000000，计数器通道 counter_set=00

        lw    $a1, 0($v1);              //再读 GPIO 端口 F0000000 状态
        and $t3,$a1,$t0;                //取最高位=out0，屏蔽其余位送 r11
//      beq       $t3,$t0,C_init;       //out0=0,Counter 通道 0 溢出,转计数器初始化,修改 7 段码显示:C_init
        add $t5, $t5, $v0;              //程序计数延时
```

```
    beq $t5, $zero,C_init;          //程序计数 r13=0,转计数器初始化,修改 7 段码显
示:C_init

l_next:                             // 判断 7 段码显示模式：SW[4:3]控制
    lw   $a1, 0($v1);               //再读 GPIO 端口 F0000000 开关 SW 状态
    add $s2, $t6, $t6;              //r14=4,r18=00000008
    add $s6, $s2, $s2;              //r22=00000010
    add $s2, $s2, $s6;              //r18=00000018(00011000)
    and $t3, $a1, $s2;              //取 SW[4:3]
    beq $t3, $zero, L20;            //SW[4:3]=00,7 段显示"点"循环移位：L20，SW0=0
    beq $t3, $s2, L21;              //SW[4:3]=11，显示七段图形，L21，SW0=0
    add $s2, $t6, $t6;              //r18=8
    beq $t3, $s2, L22;              //SW[4:3]=01,七段显示预置数字，L22，SW0=1
    sw   $t1, 0($a0);               //SW[4:3]=10，显示 r9，SW0=1
    j     loop2;

L20:
    beq $t2, $at, L4;               //r10=ffffffff,转移 L4
    j     L3;

L4:
    nor  $t2, $zero, $zero;         //r10=ffffffff
    add $t2, $t2, $t2;              //r10=fffffffe

L3:
    sw   $t2, 0($a0);               //SW[4:3]=00,7 段显示点移位后显示
    j     loop2;

L21:
    lw   $t1, 60($s1);              //SW[4:3]=11，从内存取预存七段图形
    sw   $t1, 0($a0);               //SW[4:3]=11，显示七段图形
    j     loop2;

L22:
    lw   $t1, 20($s1);              //SW[4:3]=01，从内存取预存数字
    sw   $t1, 0($a0);               //SW[4:3]=01,七段显示预置数字
    j     loop2;

C_init:
    lw   $t5, 14($zero);            //取程序计数延时初始化常数
    add $t2, $t2, $t2;              //r10=fffffffc，7 段图形点左移
    or    $t2, $t2, $v0;            //r10 末位置 1，对应右上角不显示
    add $s1, $s1, $t6;              //r17=00000004，LED 图形访存地址+4
    and $s1, $s1, $s4;              //r17=000000XX，屏蔽地址高位，只取 6 位
```

```
        add $t1, $t1, $v0;              //r9+1
        beq $t1, $at, L6;               //若 r9=ffffffff,重置 r9=5
        j     L7;

L6:
        add $t1, $zero, $t6;            //r9=4
        add $t1, $t1, $v0;              //重置 r9=5

L7:
        lw    $a1, 0($v1);              //读 GPIO 端口 F0000000 状态
        add $t3, $a1, $a1;
        add $t3, $t3, $t3;              //左移 2 位将 SW 与 LED 对齐，同时 D1D0 置 00，选择计数器通道 0
        sw   $t3, 0($v1);               //r5 输出到 GPIO 端口 F0000000，计数器通道 counter_set=00
        sw   $a2,4($v1);                //计数器端口:F0000004，送计数常数 r6=F8000000

        j     l_next;

        ……                            //从此处-0000FFFC 填“00000000”。

#DataAddre: 00001000;                   //数据地址，此处 00001000H 开始定义数据
Data1:                                  //数据区 1，标号
dd FFFFFF00, 000002AB, 80000000, 0000003F, 00000001, FFFF0000, 0000FFFF,   80000000,
00000000, 11111111, 22222222, 33333333, 44444444, 55555555,   66666666, 77777777,
88888888, 99999999, AAAAAAAA, BBBBBBBB, CCCCCCCC, DDDDDDDD, EEEEEEEE, FFFFFFFF;
db 0x55,0x56,0x57,0x58;                 //db伪指令,定义数据0x55…在00001060
dw 'ab',0x1234;                         /dw伪指令,定义数据0x41,0x42,0x1234在000010604
dd 0x12345678;                          //dd伪指令，定义数据0x12345678在00001068
data2:RESB16;                           //data2是标号，从0000106C开始定义16个字节空间

        ……          //此处 00001070-00001FFC 填“00000000”。

#DataAddre: 00002000;                   //数据地址，此处 00002000H 开始定义数据
Data2:                                  //数据区 2，标号
557EF7E0, D7BDFBD9, D7DBFDB9, DFCFFCFB, DFCFBFFF, F7F3DFFF, FFFFDF3D, FFFF9DB9,
FFFFBCFB, DFCFFCFB, DFCFBFFF, D7DB9FFF, D7DBFDB9, D7BDFBD9, FFFF07E0, 007E0FFF,
03bdf020, 03def820, 08002300;

        ……      //此处从0000204C-0000BFFC填“00000000”。
```

#DataAddre: 0000C000; //数据地址，此处0000C000H开始定义数据
buffer:RESD32; //数据区3，buffer标号=0000C000，定义32个32
位字空间

//到 0000C080 结束