一、填空

- (a)将10进制数35转换为8位二进制数是00100011
- (b)将二进制数00010101转换为16进制数是0x15
- (c)将10进制数-35转换为8位二进制补码是11011101
- (d)将8进制数204转换为10进制数是132
- (e)请判断以下两个补码表示的二进制数做二进制加法后是否溢出：是
  - 01101110
  - 00011010
  - 10001000

一、填空

❖(f)对8位16进制数0x88做符号扩展成16位数是：0xFF88

❖(g)下列代码段存储在内存中，起始地址为0x00012344，分支指令执行后PC的两个可能的值分别是：0x00012344和0x00012354。同时，请在注释位置用伪代码形式对每条指令做出描述。

```
➢ loop:   lw     $t0, 0($a0)      # t0 = mem[a0]
➢         addi   $a0, $a0, 4      # a0 = a0 +4
➢         andi   $t1, $t0, 1      # t1 = t1 & 1 “Extract LSD”
➢         beqz   $t1,  loop       # if  t0 is an even go to loop
```

二、

❖将下列汇编语言指令翻译成机器语言代码，以16进制表示

loop:	addu	$a0, $0, $t0	# 0x00082021
	ori	$v0, $0, 4	# 0x34020004
	syscall		# 0x0000000C
	addi	$t0, $t0, -1	# 0x2108FFFF
	bnez	$t0, loop	# 0x1500FFFB
	andi	$s0, $s7, 0xffc0	# 0x32F0FFC0
	or	$a0, $t7, $s0	# 0x01F02025
	sb	$a0, 4($s6)	# 0xA2C40004
	srl	$s7, $s7, 4	# 0x0017B902

三、

❖写一个MIPS汇编程序，要求对内存以“example100”为标签(label)的数据段中前100个字(words)的数据求和，并将结果存入紧跟在这100个字之后的内存中。

➢伪代码：

```
$a0 = &example100;                # “&” means “Address of”
$t0 = 0;
for ($t1= 100;  $t1 > 0;  $t1= $t1- 1)
{
      $t0 = $t0 + mem($a0);
      $a0 = $a0 + 4;
}
mem($a0) = $t0;
```

三、

Label	*Op-Code*	*Dest. S1, S2*	*Comments*
	.data		
example100:	.space	400	
	.globl	main	
	.text		
main:			
	la	$a0, example100	# Load address pointer
	li	$t0, 0	# Clear sum
	li	$t1, 100	# Initialize loop count
loop:			
	lw	$t2, 0($a0)	# $t2 = mem(a0)
	add	$t0, $t0, $t2	# $t0 = $t0 + $t2
	addi	$a0, $a0, 4	# Inc. address pointer
	addi	$t1, $t1, -1	# Dec. loop count
	bgtz	$t1, loop	# if ($t1 > 0) branch
	sw	$t0, 0($a0)	# Store the result
	li	$v0, 10	# End of program
	syscall		

四、

❖写一段MIPS汇编语言代码，将内存中“SRC”标签开始的100个字的一块数据转移到内存中另一块以“DEST”标签开始的空间中。

➢伪代码：

```
$a1= &SRC;                    # “&” means “Address of”
$a2= &DEST;
for ($t0 = 100; $t0 > 0; $t0 =$t0 -1)
{
    $t1 = mem($a1);
    mem($a2) = $t1;
    $a1= $a1 + 4;
    $a2= $a2 + 4;
}
```

四、

Label	*Op-Code*	*Dest. S1, S2*	*Comments*
	.data		
SRC:	.space	400	
DEST:	.space	400	
	.globl	main	
	.text		
main:			
	la	$a1, SRC	#$a1 = &SRC
	la	$a2, DEST	#$a2 = &DEST
	li	$t0, 100	#$t0 = 100
loop:	lw	$t1, 0($a1)	#$t1= mem($a1)
	sw	$t1, 0($a2)	#mem($a2) = $t1
	addi	$a1, $a1,4	#$a1 = $a1+4
	addi	$a2, $a2,4	#$a2 = $a2+4
	addi	$t0, $t0,-1	#$t0 = $t0 - 1
	bgtz	$t0, loop	#Branch if $t0 > 0
	li	$v0, 10	
	syscall		

五、

❖写一个MIPS函数ABS，通过$a0传入一个32位整数，将这个数的绝对值存回$a0。再写一段主程序，调用两次ABS并输出结果，每次传给ABS的数不同。

➢函数伪代码：

```
function ABS($a0);
if ($a0 < 0)
        $a0 = $0 - $a0;
return;
```

五、

```
Label     Op-Code   Dest. S1, S2      Comments
          .text
ABS:      bgez      $a0, return       # if ($a0 >= 0) done
          sub       $a0, $0, $a0      # $a0 = 0 - $a0
return:   jr        $ra               # return
##################################################
          .globl    main
          .text
main:     li        $a0, -8765
          jal       ABS
          li        $v0, 1            # Output result
          syscall
          li        $a0, 4321
          jal       ABS
          li        $v0, 1            # Output result
          syscall
          li        $v0, 10           # End of program
          syscall
```

六、

❖写一个函数FIB(N, &array)向内存中的一个数组(array)存入斐波那契数列的前N个元素。N和array的地址分别通过$a0和$a1传递进来。斐波那契数列的前几个元素是：1, 1, 2, 3, 5, 8, 13, ……

➢伪代码：

```
mem($a1) = 1;
mem($a1 + 4) = 1;
for ($a0 = $a0 - 2; $a0 > 0; $a0 = $a0-1)
{
    mem($a1+8) = mem($a1) + mem($a1+4);
    $a1 = $a1 + 4;
}
return;
```

六、

Label	*Op-Code*	*Dest. S1, S2*	*Comments*
fib:	li	$t0, 1	
	sw	$t0, 0($a1)	
	sw	$t0, 4($a1)	
	addi	$a0, $a0, -2	
loop:			
	lw	$t0, 0($a1)	
	lw	$t1, 4($a1)	
	add	$t0, $t0, $t1	
	sw	$t0, 8($a1)	
	addi	$a1, $a1, 4	
	addi	$a0, $a0, -1	
	bgtz	$a0, loop	
	jr	$ra	

七、

❖写一个函数，从$a0、 $a1和$a2中接受传递过来的3个32位整数，按从小到大排序后存回$a0~$a2。

➢伪代码：

```
function order($a0,$a1,$a2);
if ($a0 > $a1)
    exchange $a0 and $a1;
if ($a1 > $a2)
    exchange $a1 and $a2
else
    return;
if ($a0 > $a1)
    exchange $a0 and $a1;
return;
```

七、

Label	*Op-Code*	*Dest. S1, S2*	*Comments*
	.text		
order:			
	ble	$a0, $a1, next	
	move	$t0, $a1	
	move	$a1, $a0	
	move	$a0, $t0	
next:			
	ble	$a1, $a2, done	
	move	$t0, $a2	
	move	$a2, $a1	
	move	$a1, $t0	
	ble	$a0, $a1, done	
	move	$t0, $a1	
	move	$a1, $a0	
	move	$a0, $t0	
done:	jr	$ra	