计算机组成原理作业一

2021141450109 胡展鹏

一、

2.4

```
int main(){
    int *x,*y;
    x=&A[f],y=&B[g];
    f=A[f];
    *y=f+*(x+1);
}
```

2.5

```
sll $t0,$s0,2
add $t0,$s6,$t0
sll $t1,$s1,2
add $t1,$s7,$t1
lw $s0,0($t0)
lw $t0,4($t0)
add $t0,$t0,$s0
sw $t0,0(t1)
```

2.6

1)

```
tmp1=Array[0];
tmp2=Array[1];
Array[1]=tmp1;
Array[0]=Array[4];
Array[4]=Array[3];
Array[3]=tmp2;
```

2)

```
lw $t0,0($s6)
lw $t1,4($s6)
sw $t0,4($s6)
sw 16($s6),4($s6)
sw 12($s6),16($s6)
sw $t1,12($s6)
```

2.7

Little-endian:

Address	Data Byte	Binary
0	0x12	0001 0010
4	0xef	1110 1111
8	0xcd	1100 1101
12	0xab	1010 1011

Big-endian:

Address	Data Byte	Binary
0	0xab	1010 1011
4	0xcd	1100 1101
8	0xef	1110 1111
12	0x12	0001 0010

2.9

```
sll $t0,$s3,2
add $t0,$t0,$s6
lw $t0,0($t0)
sll $t1,$s4,2
add $t1,$t1,$s6
lw $t1,0($t1)
add $t2,$t0,$t1
sw $t2,32($s7)
```

2.10

```
int main(){
    A[1]=&A[0];
    &A[1]=A[1];
    f=&A[1]+&A[0];
}
```

二、

2.11

节选

R-type	op	rs	rt	rd
add	000000	rs	rt	rd

R-type	op	rs	rt	rd
sub	000000	rs	rt	rd
and	000000	rs	rt	rd
or	000000	rs	rt	rd
slt	000000	rs	rt	rd
sll	000000	00000	rt	rd
jr	000000	rs	00000	00000
I-type	**op**	**rs**	**rt**	**immediate**
addi	001000	rs	rt	immediate
andi	001100	rs	rt	immediate
ori	001101	rs	rt	immediate
lui	001111	00000	rt	immediate
lw	100011	rs	rt	immediate
sw	101011	rs	rt	immediate
beq	000100	rs	rt	immediate
J-type	**op**	**address**	**address**	**address**
j	000010	address	address	address
jal	000011	address	address	address

the immediate field: $0 \sim 2^{16} - 1$

the destination register field: $0 \sim 2^{26} - 1$

2.12

1. 0x50000000
2. there has been overflow
3. 0xB0000000
4. no overflow
5. 0xD0000000
6. there has been overflow

2.13

1. $x > 2^{31} - 129$
2. $x < -2^{31} + 129$
3. $x < -2^{31} + 128$

2.14

r-type, add $s0,$s0,$s0

2.15

i-type, 0xAD490020

2.16

r-type, sub $v1,$v1,$v0

0x00621822

2.17

i-type, lw $v0,4($at)

0x8C220004

2.19

1.
2. 0xAAAAAAA0
3. 0x00005545

三、

2.20

```
srl $t0,$t0,11
sll $t0,$t0,26
ori $t2,$0,0x03ff
sll $t2,$t2,16
ori $t2,$t2,0xffff
and $t1,$t1,$t2
or $t1,$t1,$t0
```

2.21

```
nor $t1,$t2,$t2
```

2.25

1. i-type
2. ```
   addi $t2,$t2,-1
   beq $t2,$0,loop
   ```

2.26

1. 20
   ```

2.
```
int main(){
    int i=10;
    do{
        B+=2;
        i--;
    }while(i>0);
}
```

3. 5*N

2.27

```
        addi $t0,$0,0
        beq $0,$0,TEST1
LOOP1:  addi $t1,$0,0
        beq $0,$0,TEST2
LOOP2:  add $t3,$t0,$t1
        sll $t2,$t1,4
        add $t2,$t2,$t2
        sw $t3,($t2)
        addi $t1,$t1,1
TEST2:  slt $t2,$t1,$s1
        bne $t2,$0,LOOP2
        addi $t0,$t0,1
TEST1:  slt $t2,$t0,$s0
        bne $t2,$0,LOOP1
```

2.28

14 instructions to implement and 158 instructions executed

2.31

```
fib:    addi $sp,$sp,-12
        sw $ra,8($sp)
        sw $s0,4($sp)
        sw $a0,0($sp)
        bgt $a0,$0,test2
        add $v0,$0,$0
        j rtn
test2:  addi $t0,$0,1
        bne $t0,$a0,gen
        add $v0,$0,$t0
        j rtn
gen:    subi $a0,$a0,1
        jal fib
        add $s0,$v0,$0
        sub $a0,$a0,1
        jal fib
        add $v0,$v0,$s0
rtn:    lw  $a0,0($sp)
        lw $s0,4($sp)
        lw $ra,8($sp)
        addi $sp,$sp,12
```

```
        jr $ra
```

2.34

```
f:  addi $sp,$sp,-12
    sw $ra,8($sp)
    sw $s1,4($sp)
    sw $s0,0($sp)
    move $s1,$a2
    move $s0,$a3
    jal func
    move $a0,$v0
    add $a1,$s0,$s1
    jal func
    lw $ra,8($sp)
    lw $sl,4($sp)
    lw $s0,0($sp)
    addi $sp,$sp,12
    jr $ra
```