2.4：

B[g]=A[f]+A[f+1]

2.8：

f=2*A

2.17：

1:

sll $t2,$t0,44 =sll $t2,$t0,12

$t2=0xAAAAA000,

or $t2,$t2,$t1 //0xAAAAA000| 0x12345678

$t2=0xBABEF678

2:

sll $t2,$t0,4 //$t2=0xAAAAAAA0

Since -1 in two`s complement is 0xFFFFFFFF

addi $t2,$t2,-1 //0xAAAAAAA0 & 0xFFFFFFFF

$t2=0xAAAAAAA0

3:

0xAAAAAAAA shift 3 bit is 0x55555550

addi $t2,$t2,0xFFEF //0x55555550 & 0x0000FFEF

$t2=0x00005545

2.18:

srl $t0,$t0,11

sll $t0,$t0,26

//Get the bit 16 down to bit 11 and put them into bit 31 down to bit 26

addi $t2,$t1,0x03FFFFF

//Remove the bit 31 to bit 26

or $t1,$t2,$t0

2.21

$t2=3

2.23

1: I format

2:

Assume that x29 store in $t0

```
        blt   $t1,$t0,$0
Loop:   beq   $t1,$0,exit
        subi  $t1,$t1,1
        J     loop
Exit:
```

2.24

1:$s2=20

2:

```
int i=10,B=0;
for(;i>0;i--){
      B=   B+2;
}
```

3:5N+2

2.25

```
            li    $t0, 0              //initialize i =0
loop_I:     bge   $t0, $s0, exit_i    //exit when i>=a
            li    $t1,0               //initialize j=0
loop_j:     bge   $t1, $s1, exit_j    //exit when j>=b
            sll   $t2, $t1, 2         //get j*4
            add   $t2, $s2, $t2       //$t2=D[4*j]
            add   $t3,$t1,$t0         //get j+i
            sw    $t3, 0($t2)         //save j+i to $t3
            addi  $t1, $t1, 1         //j++
            j     loop_j
exit_j:     addi  $t0, $t0, 1
            j     loop_i
exit_i:
```

2.27

```
for(i=0;i<100;i++){
      Result=result+memArray[i];
}
```

2.28

```
            addi  $t0, $s0, 400       //get the &MemArray[100]
Loop:       lw    $s1, 0($s0)         //get the &MemArray[0]
            add   $s1,$s1,$s2         //result+MemArray[i]
            addi  $s0, $s0, 4         //$s1=$MemArray[i+1]
            bne   $s0,$t0,loop
```

2.39

1:

For the first situation: $n_{cycle}=n_{arithmetic}\times 1+n_{load/store}\times 10+n_{branch}\times 3$

$=500\times 1+300\times 10+100\times 3$

$=3800$

$T_{run}=t_{cycle}\times n_{cycle}=3800\times t_{cycle}$

For the second situation: n_{cycle} $=n_{arithmetic}\times 1\times 75\%+n_{load/store}\times 10+n_{branch}\times 3$

$=500\times 1\times 0.75+300\times 10+100\times 3$

$=3675$

T_{run} $=t_{cycle}\times n_{cycle}$

$=1.1\times t_{cycle}\times 3675$

$=4042.5t_{cycle}$

So, it is not a good design

2:

$n_{cycle}=n_{arithmetic}\times 1\times 50\%+n_{load/store}\times 10+n_{branch}\times 3=3550$

speedUp is $\frac{3800}{3550}\approx 1.07$

$n_{cycle}=n_{arithmetic}\times 1\times 10\%+n_{load/store}\times 10+n_{branch}\times 3=3350$

speedUp is $\frac{3800}{3350}\approx 1.13$

2.40

1:

CPI=70%×2+10%×6+20%×3=2.6

2:

2.6×75%=1.95

Assume x cycle arithmetic instruction takes

1.95=70% x+10%×6+20%×3

X =1.07

3:

2.6×50%=1.3

Assume x cycle arithmetic instruction takes

1.3=70% x+10%×6+20%×3

X =0.14