Assignment 2
Xiaohui Chen (xc2388)

4.43
A. It doesn't correctly decribes pushl %esp. When push %esp is executed, an old value of %esp(not decremented by 4) is pushed into the stack. The code sequences will first decrement 4 to %esp and then push it to the stack.

B.
movl REG, %eax
subl $4,%esp
movl %eax, (%esp)

4.44
A.
No. Because when popl %esp executed, the original value %esp points to will stored as %esp. Instead, %esp will not simply increment 4.
B.
movl (%esp),%eax
addl $4,%esp
movl %eax,REG

4.45
A.

```
#include<stdio.h>
void bubble_a(int *data, int count);
int main()
{
   int arr[10]={8,10,6,4,9,3,7,1,5,2};
   int *p=arr;
   int count=10;
   int k;
   bubble_a(p,10);
   for(k=0;k<count;k++)
   {
       printf("%d\n",arr[k]);
   }
   return 0;
}

void bubble_a(int *data ,int count)
{
    int i,last;
    for(last=count-1;last>0;last--)
    {
      i=0;
      while(i<last)
      {
          int *tmp=data+i;
          int *tmp1=data+i+1;
```

```
        i+=1;
        if(*tmp>*tmp1)
        {
           int t=*tmp;
           *tmp=*tmp1;
           *tmp1=t;
         }
     }
    }
}
```

B.

```
      .pos0
init: irmovl Stack,%esp
       irmovl Stack,%esp
       call Main
       halt
 # the array is {8,10,6,4,9,3,7,1,5,2}
# result should be {1,2,3,4,5,6,7,8,9,10}
Main:   pushl %ebp
          rrmovl %esp, %ebp
         irmovl $8,%edx
          pushl %edx
          irmovl $10,%edx
          pushl %edx
          irmovl $6,%edx
          pushl %edx
          irmovl $4,%edx
          pushl %edx
          irmovl $9,%edx
          pushl %edx
          irmovl $3,%edx
          pushl %edx
          irmovl $7,%edx
          pushl %edx
          irmovl $1,%edx
          pushl %edx
          irmovl $5,%edx
          pushl %edx
          irmovl $2,%edx
          pushl %edx
          irmovl $10,%eax
          irmovl $40,%ecx
          rrmovl %esp, %ebx
          addl %ecx,%ebx

          call Bubble
          ret

Bubble:  pushl %ebp
           rrmovl %esp,%ebp
           #pushl %ebx
```

```
            irmovl $1,%ecx
            subl %ecx,%eax                          #last
            jle End

Loop:  irmovl $0,%edx                       #i
          rrmovl %eax,%ecx
          subl %edx,%ecx
          jle End1

Loop11:  irmovl $4,%ecx
         subl %ecx,%ebx
         mrmovl (%ebx),%esi
         mrmovl -4(%ebx), %edi
         subl %esi,%edi
         jge  Loop12

Swap:     addl %esi,%edi
          rmmovl %edi,(%ebx)
          rmmovl %esi,-4(%ebx)

Loop12:        irmovl $1,%ecx
               addl %ecx,%edx
               rrmovl %edx,%ecx
               subl %eax,%ecx
               jl Loop11

End1:          irmovl $48,%ecx
              rrmovl %esp,%ebx
              addl %ecx,%ebx
              irmovl $1,%ecx
              subl %ecx,%eax
              jg Loop

End:      ret

         .pos 0x400
Stack:
```

4.47

```
Fetch       icode:ifun<----M1[PC]    (icode=C    ifun=0)
            rA:rB<----M1[PC+1]
            valC<----M1[PC+2]        (valC=V)
            valP<----PC+6

Decode    valB<----R[rB]

Execute   valE<----valC+valB         (valE=V+valC)

Write back    R[rB]<----valE

PC update     PC<----valP
```

4.48

Fetch	icode:ifun<----M1[PC]	(icode=D ifun=0)
	rA:rB<----M1[PC+1]	
	valP<----PC+1	
Decode	valA<----R[%ebp]	
	valB<----R[%ebp]	
Execute	valE<----valB+4	
Memory	valM<----M4[valA]	
Write back	R[%esp]<----valE	
	R[%ebp]<----valM	
PC update	PC<----valP	