

一. 针对如下 C 程序及其在 i386 Linux 下的汇编代码（片段）:

```
#include<stdio.h>
union var{
    char c[5];
    int i;
};

int main(){

union var data;
char *c;

data.c[0] = '2';
data.c[1] = '0';
data.c[2] = '1';
data.c[3] = '6';
data.c[4] = '\0';

c = (char*)&data;

printf("%x %s\n",data.i,c)
;

return 0;
}

//第一题 C 程序
```

```
.section .rodata
.LC0:
.string "%x %s\n"
.text
.globl main
.type main,@function
main:

movl %esp,%ebp
subl $40,%esp
andl $-16,%esp
movl $0,%eax
subl %eax,%esp
movb $50,-24(%ebp)

movl %eax,-28(%ebp)

pushl $.LC0
call printf
addl $16,%esp

leave
ret

//第一题 汇编程序
```

- (a) 上述 C 程序的输出是什么?
- (b) 补全 10 处划线部分的汇编代码。

二. 针对如下 C 程序及其汇编代码（片段）：

```

#define N 2
// #define N 11
typedef struct POINT {
    int x, y ;
    char z[ N ];
    struct POINT *next;
} DOT;
void f(DOT p)
{
    p.x = 100;
    p.y = sizeof(p);
    p.z[1] = 'A';
    f(* (p.next));
} //第二题 C 程序

```

```

.file "test1.c"
.text
.globl f
.type    f,@function
f:
    pushl   %ebp
    movl    %esp,    %ebp
    movl    $100,    8(%ebp)
    movl    $16,    12(%ebp)
    movb    $65,    _____, %eax
    pushl   _____
    pushl   _____
    pushl   _____
    pushl   _____
    call    f
    addl    $16,    %esp
    leave
    ret

```

//当 N=2 时，生成的汇编代码片段。

```

.file "test1.c"
.text
.globl f
.type    f,@function
f:
    pushl   %ebp
    movl    %esp,    %ebp
    pushl   %edi
    pushl   %esi
    movl    $100,    8(%ebp)
    movl    _____, 12(%ebp)
    movb    $65,    _____
    subl    $8,    %esp
    movl    _____, %eax
    subl    $24,    %esp
    movl    %esp,    %edi
    movl    %eax,    %esi
    cld
    movl    _____, %eax
    movl    %eax,    %ecx
    rep
    movsl
    call    f
    addl    $32,    %esp
    leal   -8(%ebp), %esp
    popl   %esi
    popl   %edi
    leave
    ret

```

// rep movsl 为数据传送指令，即，由源地址 esi 开始的 ecx 个字的数据传送到由 edi 指示的目的地址。

//当 N=11 时，生成的汇编代码片段

- (a) 补全划线处的汇编代码；
- (b) 从运行时环境看，`addl $16, %esp` 和 `leal -8(%ebp), %esp` 这两条汇编指令的作用是什么？
- (c) 结合上述两种汇编代码，简述编译器在按值传递结构变量时的处理方式。

三. 针对如下 C 程序及其汇编代码 (片段):

```
void g(int**);
int main()
{
    int line[10], i;
    int *p=line;
    for (i=0; i<10; i++)
    { *p=i; g(&p); }
    return 0;
}
void g(int**p)
{ (**p)++; (*p)++; }
//第三题 c 程序
```

(a) 补全下划线处的空白汇编代码;
 (b) main 函数中 for 循环结束时,
 数组 line 各元素值是多少?

```
.globl g
.type    g,@function
g:
    pushl   %ebp
    movl   %esp, %ebp
    movl   _____, %eax
    movl   _____, %eax
    _____
    movl   _____, %eax
    _____
    leave
    ret
```

//第三题函数 g 的汇编代码片段

```
.file "p.c"
.text
.globl main
.type    main,@function
main:
    pushl   %ebp
    movl   %esp, %ebp
    subl   $72, %esp
    andl   $-16, %esp
    movl   $0, %eax
    subl   %eax, %esp
    leal   -56(%ebp), %eax
    movl   %eax, -64(%ebp)
    movl   $0, -60(%ebp)
```

.L2:

①

jle .L5
 jmp .L3

.L5:

```
    movl   -64(%ebp), %edx
    movl   -60(%ebp), %eax
    movl   %eax, (%edx)
    subl   $12, %esp
    leal   -64(%ebp), %eax
    pushl   %eax
    call   g
```

②

```
    leal   -60(%ebp), %eax
    incl   (%eax)
```

③

.L3:

```
    movl   $0, %eax
    leave
    ret
```

//第三题函数 main 的汇编代码片段

四. 针对如下 C 程序及其汇编代码 (片段):

- (1) 补全下划线处的空白汇编代码;
- (2) 描述所用编译器对 C 分程序所声明变量的存储分配策略;

```
#include <stdio.h>
int main()
{
    int a=0, b = 0;
    { int a = 1; }
    { int b = 2;
      { int a = 3; }
    }
    return 0;
} //第四题 C 程序
```

```
main:
    pushl %ebp
    _____
    subl $24, %esp
    andl $-16, %esp
    movl $0, %eax
    subl %eax, %esp
    movl $0, _____
    movl $0, _____
    movl $1, _____
    movl $2, -12(%ebp)
    movl $3, _____
    movl $0, %eax
    leave
    ret
//第四题 汇编代码片段
```

```
#include <stdio.h>
int main()
{
    int a[6]={0,1,2,3,4,5};
    int i=6,j=7;
    int *p = (int*)(&a+1);
    printf("%d\n",*(p-1));
    return 0;
} //第五题 C 程序
```

```
.LC0:
    .long 0
    .long 1
    .long 2
    .long 3
    .long 4
    .long 5
.LC1:
    .string "%d\n"
    .text
.globl main
.type main,@function
main:
    pushl %ebp
    movl %esp, %ebp
    pushl %edi
    pushl %esi
    subl $48, %esp
    andl $-16, %esp
    movl $0, %eax
    subl %eax, %esp
    leal -40(%ebp), %edi
    movl $.LC0, %esi
    cld
    movl $6, %eax
    movl %eax, %ecx
    rep
    movsl
    movl $6, -44(%ebp)
    movl $7, -48(%ebp)
    leal -40(%ebp), %eax
    addl _____
    movl %eax, -52(%ebp)
    subl $8, %esp
    movl -52(%ebp), %eax
    subl $_____, %eax
    pushl _____
    pushl $.LC1
    call printf
    addl $_____, %esp
    movl $0, %eax
    leal _____, %esp
    popl %esi
    popl %edi
    leave
    ret
```

//第五题 汇编代码片段

五. 仔细阅读所给 C 程序及其汇编代码片段。

- (1) 指出波浪线处的汇编代码的作用;
- (2) 补全下划线处的空白汇编代码。

六. 假设以下假想的程序采用静态嵌套作用域规则:

```
program staticLink
  procedure f(level, arg())
    //函数 f 有两个参数, 整型变量 level, 无参函数 arg

    procedure local() // 嵌套在 f 中的函数
      begin //无参函数 local, 返回一个整型值。
        return level;
      end

  begin //f 函数体
    if (level > 10) return f(level-1, local);
    else if (level > 1) return f(level-1, arg); else return arg();
  end

  procedure dummy()
  begin /*空的函数体*/end

begin //staticLink 函数体
  print(f(17, dummy));
end
```

- (a) 给出该程序运行结果;
- (b) 给出函数调用 `f(17, dummy)` 执行时运行栈上包含活动记录最多时的相关图示。假设按照逆序方式传递参数; 函数作为参数传递时, 需要两个单元, 一为函数入口地址 (可用函数名表示), 二为其访问链。