

CMPE 230 Systems Programming Homework 3

Problem 1

Write a CGI program called `last100.cgi` which prints the names and IP numbers of the last 100 persons who visited the page. The name of the person will be input to the program using the GET method. No forms will be used. A user can invoke your cgi program either as follows:

```
http://localhost/cgi-bin/last100.cgi?name=ali
```

or simply as :

```
http://localhost/cgi-bin/last100.cgi
```

In the first case, the name `ali` and its IP number should also be added to the list of visitors and the 100 name and IP number pairs should be printed. In the second case, no name should be added and just the previous 100 name and IP number pairs should be printed.

Problem 2

Write an A86 assembly language program which will input strings separated by a blank character. Assume the strings contain just alphanumeric characters. The first string will be searched for a substring given by the second string. If the substring occurs in the first string, then `y` should be printed. If it does not occur, then `n` should be printed as output.

Problem 3

The following code is A86 assembly language code. It prints the number 255 (stored at the location labelled as `number`) using hexadecimal and binary representation. The output of the program is:

```
FF 11111111
```

The program basically works by repeatedly dividing the number by the base and pushing the remainder on the stack. The contents of the stack is then popped to print the digits in correct order. Fill in the missing places in the program.

```
code segment
    mov     -----,-----                /* (a,b) */
    mov     di,2          ; iterate formatloop twice(hex and binary)
    mov     si,10h       ; divide by 16 for hex format
formatloop:
    mov     ax,[bx]
    xor     dx,dx
    push   20h          ; push blank
    mov     cx,1d       ; character counter on stack
more:
```

```

    div    si
    cmp    dl,_____ /* (c) */
    jb     _____ /* (d) */
letter:
    sub    dl,_____ /* (e) */
    add    dl,_____ /* (f) */
    jmp    enddigit
digit:
    add    dl,_____ /* (g) */
enddigit:
    push  dx
    inc   cx
    xor   dx,dx
    cmp   ax,0h
    jne   more
writeloop:      ; pop and display characters
    pop   dx
    mov   ah,02h
    int   21h
    dec   cx
    jnz   writeloop
    mov   si,2d      ; divide by 2 for binary format
    dec   di         ; check to see if we will loop
    jnz   _____ /* (h) */
progfinish:
    int   20h
number:
    dw    255d
code ends

```

Note that the instruction `jb` means jump on below.

Problem 4

The following program was generated by the GNU assembler. Disassemble the program. i.e. write down the closest C program which produces the following GNU assembly code.

```

.file "2.c"
.section .rodata
.LC0:
.string "Value is %d\n"
.text
.globl func
.type func,@function
func:
pushl %ebp
movl %esp, %ebp
subl $8, %esp

```

```

movl 12(%ebp), %eax
addl 8(%ebp), %eax
addl $10, %eax
movl %eax, -4(%ebp)
subl $8, %esp
pushl -4(%ebp)
pushl $.LC0
call printf
addl $16, %esp
leave
ret
.Lfe1:
.size func,.Lfe1-func
.globl main
.type main,@function
main:
pushl %ebp
movl %esp, %ebp
subl $8, %esp
andl $-16, %esp
movl $0, %eax
subl %eax, %esp
movl $3, -4(%ebp)
movl $4, -8(%ebp)
subl $8, %esp
pushl -8(%ebp)
pushl -4(%ebp)
call func
addl $16, %esp
leave
ret
.Lfe2:
.size main,.Lfe2-main
.ident "GCC: (GNU) 3.2.2 20030222 (Red Hat Linux 3.2.2-5)"

```

Note that the output of the program is:

```
Value is 17
```

Problem 5

Consider the C program:

```

/***** C program *****/
#include <stdio.h>

func()
{

```

```

    int a,b,c ;
    register int d ;

    a = -4 ;
    b = -8 ;
    c = a + b ;
    d = c + a ;
    /* HERE */
}

main()
{
    short x ;

    x = -4 ;
    func() ;
}

```

This C program is compiled with gcc -S example.c command on the linux which produces the following gnu assembler output:

```

.file "1.c"
.text
.globl func
.type func,@function
func:
pushl %ebp
movl %esp, %ebp
subl $12, %esp
movl $-4, -4(%ebp)
movl $-8, -8(%ebp)
movl -8(%ebp), %eax
addl -4(%ebp), %eax
movl %eax, -12(%ebp)
leave
ret
.Lfe1:
.size func,.Lfe1-func
.globl main
.type main,@function
main:
pushl %ebp
movl %esp, %ebp
subl $8, %esp
andl $-16, %esp
movl $0, %eax
subl %eax, %esp
movw $-4, -2(%ebp)

```

```
call func
leave
ret
.Lfe2:
.size main,.Lfe2-main
.ident "GCC: (GNU) 3.2.2 20030222 (Red Hat Linux 3.2.2-5)"
```

Answer the following questions about the above programs:

1. Circle and name all the references to the C variables `a`, `b`, `c`, `d` and `x` on the assembly code.
2. Draw the memory layout of the program when it reaches the point marked as `/* HERE */` during its execution. On the diagram show memory locations of all the variables and the contents of the registers (if you do not know the exact contents, you can show where they point to in the memory diagram).