Computer Workshop

LAB 3

Alireza Karimi

810101492

Professor Hosseini

Aban, 1402

First question:

Here’s my code snipped:(about writing 20 characters)

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
const int SIZE = 20;
int main()
{
    srand(time(NULL));
    char str[SIZE], inp[SIZE];

    for (int i = 0; i < SIZE; i++)
    {
        str[i] = 'a' + (rand() % 26);
        printf("%c",str[i]);
    }
    float accuracy = 0;
    printf("\nPlease enter 20 characters:\n");
    time_t start_time = time(NULL);
    for (int i = 0; i < SIZE; i++)
    {
        inp[i] = getchar();
        accuracy += (float)(inp[i] == str[i])/SIZE;
    }
    time_t end_time = time(NULL);
    time_t difftime = end_time - start_time;
    printf("Your accuracy was: %.2lf%%\nYour speed was: %lds\n",accuracy * 100, difftime);
    return 0;
}
```

My inputs and outputs:

```
(base) alireza@alirezas-MacBook-Air 3 % ./a.out
sffjraohcgfalciwmcyn
Please enter 20 characters:
sffjraohcgfalciwmcyn
Your accuracy was: 100.00%
Your speed was: 21s
(base) alireza@alirezas-MacBook-Air 3 %
```

```
(base) alireza@alirezas-MacBook-Air 3 % gcc 1.c
(base) alireza@alirezas-MacBook-Air 3 % ./a.out
zivxmwimabynyeffrkhe
Please enter 20 characters:
zuiwnwimanasffffrkhs
Your accuracy was: 50.00%
Your speed was: 17s
```

```
(base) alireza@alirezas-MacBook-Air 3 % ./a.out
ocdslwjzblmzxgihauht
Please enter 20 characters:
ocasknasan,mmsaasnak
Your accuracy was: 15.00%
Your speed was: 6s
```

```
(base) alireza@alirezas-MacBook-Air 3 % ./a.out
etxebgfmvtbisiwsfyoc
Please enter 20 characters:
askjddakndkadndnakjs
Your accuracy was: 0.00%
Your speed was: 5s
```

Second question:

It’s about creating three assembly files for multiplying 5 and 8 in three different ways:

First one: using ‘mul’ instruction:

```
 GNU nano 2.2.6                File: mul01.s                          Mo

.global main
.func main

main:
    mov r2, #5      @ Load number 5 into r0
    mov r1, #8      @ Load number 8 into r1

    mul r0, r2, r1 @ Multiply r0 and r1, and store the result in r2

    mov r7, #0      @ Set exit code to 0
    bx lr           @ Return from main
```

```
pi@raspberrypi ~/test $ gcc -o mul01 mul01.o
pi@raspberrypi ~/test $ as -o ./mul01.o ./mul01.s
pi@raspberrypi ~/test $ gcc -o mul01 mul01.o
pi@raspberrypi ~/test $ ./mul01
pi@raspberrypi ~/test $ echo $?
40
pi@raspberrypi ~/test $ _
```

Second one: using ‘shifting’

```
 GNU nano 2.2.6                File: mul02.s

.global main
.func main

main:
    mov r0, #5     @ Load number 5 into r0

    lsl r0, r0, #3 @ Shift the value in r0 3 bits to the left, and store the re$

    mov r7, #0     @ Set exit code to 0
    bx lr          @ Return from main

pi@raspberrypi ~/test $ as -o ./mul02.o ./mul02.s
pi@raspberrypi ~/test $ gcc -o mul02 mul02.o
pi@raspberrypi ~/test $ ./mul02
pi@raspberrypi ~/test $ echo $?
40
```

Third one: using ‘Adding’:

```
 GNU nano 2.2.6                File: mul03.s

.global main
.func main
main:
    mov r0, #5     @ Load number 5 into r0

    mov r4, #0     @ Initialize r4 as 0
    ldr r5, =8     @ Load number 8 into r5

    loop:
        add r4, r4, r0    @ Add r0 to r4, accumulating the result
        subs r5, r5, #1   @ Decrement r5 by 1
        cmp r5, #0        @ Compare r5 with 0
        bne loop          @ Branch to loop if r5 is not equal to 0

    mov r0, r4
    mov r7, #0     @ Set exit code to 0
    bx lr          @ Return from main


pi@raspberrypi ~/test $ as -o ./mul03.o ./mul03.s
pi@raspberrypi ~/test $ gcc -o mul03 mul03.o
pi@raspberrypi ~/test $ ./mul03
pi@raspberrypi ~/test $ echo $?
40
```

Third question:

This question is about comparing two numbers in assembly:

I got the outputs in qemu

```
.global main
.func main

.data
num1:   .word 5         @ First number stored in RAM
num2:   .word 7         @ Second number stored in RAM

main:
    ldr r1, =num1       @ Load the address of the first number into r1
    ldr r1, [r1]        @ Load the value of the first number into r1

    ldr r2, =num2       @ Load the address of the second number into r2
    ldr r2, [r2]        @ Load the value of the second number into r2

    cmp r1, r2          @ Compare the values of the two numbers
    movgt r0, #1        @ If r1 > r2, move 1 into r0 (first number is bigger)
    movle r0, #2        @ If r1 <= r2, move 2 into r0 (second number is bigger)
    bx lr               @ Return from the function
```

```
pi@raspberrypi ~/test $ as -o ./compare01.o ./compare01.s
pi@raspberrypi ~/test $ gcc -o compare01 compare01.o
pi@raspberrypi ~/test $ ./compare01
pi@raspberrypi ~/test $ echo $?
2
```

```
.global main
.func main

.data
num1:   .word 5         @ First number stored in RAM
num2:   .word 3         @ Second number stored in RAM

main:
    ldr r1, =num1       @ Load the address of the first number into r1
    ldr r1, [r1]        @ Load the value of the first number into r1

    ldr r2, =num2       @ Load the address of the second number into r2
    ldr r2, [r2]        @ Load the value of the second number into r2

    cmp r1, r2          @ Compare the values of the two numbers
    movgt r0, #1        @ If r1 > r2, move 1 into r0 (first number is bigger)
    movle r0, #2        @ If r1 <= r2, move 2 into r0 (second number is bigger)
    bx lr               @ Return from the function
```

```
pi@raspberrypi ~/test $ as -o ./compare01.o ./compare01.s
pi@raspberrypi ~/test $ gcc -o compare01 compare01.o
pi@raspberrypi ~/test $ ./compare01
pi@raspberrypi ~/test $ echo $?
1
```

Fourth question:

About Disassembling:

First one:

```c
int main()
{
    int num1 = 5, num2 = 7;
    return (num1 > num2) ? 1 : 2;
}
```

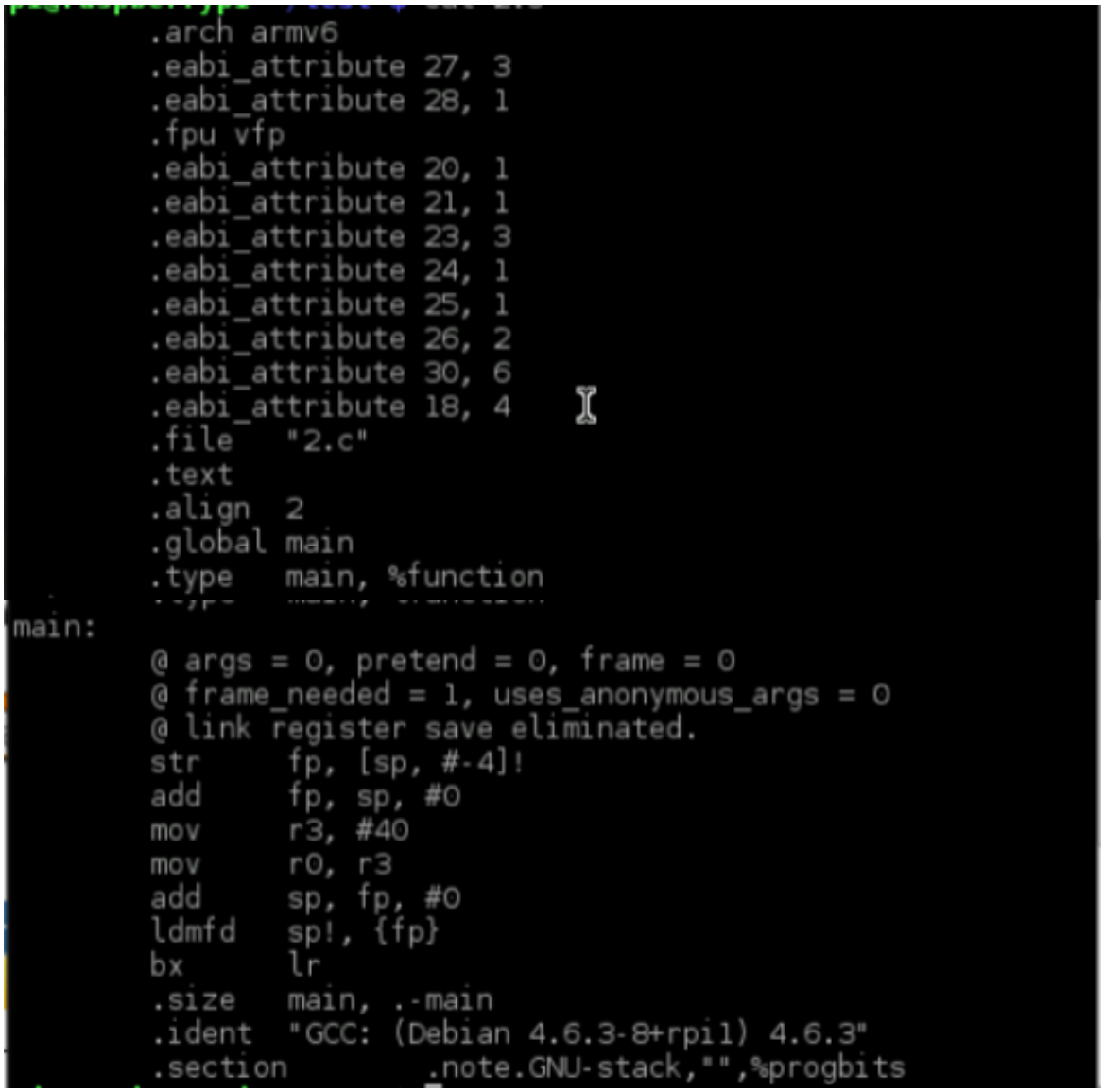
Disassembled code:

```
pi@raspberrypi ~/test $ cat 1.s
        .arch armv6
        .eabi_attribute 27, 3
        .eabi_attribute 28, 1
        .fpu vfp
        .eabi_attribute 20, 1
        .eabi_attribute 21, 1
        .eabi_attribute 23, 3
        .eabi_attribute 24, 1
        .eabi_attribute 25, 1
        .eabi_attribute 26, 2
        .eabi_attribute 30, 6
        .eabi_attribute 18, 4
        .file   "1.c"
        .text
        .align  2
        .global main
        .type   main, %function
main:
        @ args = 0, pretend = 0, frame = 0
        @ frame_needed = 1, uses_anonymous_args = 0
        @ link register save eliminated.
        str     fp, [sp, #-4]!
        add     fp, sp, #0
        mov     r3, #40
        mov     r0, r3
        add     sp, fp, #0
        ldmfd   sp!, {fp}
        bx      lr
        .size   main, .-main
        .ident  "GCC: (Debian 4.6.3-8+rpi1) 4.6.3"
        .section        .note.GNU-stack,"",%progbits
```

Second one:

```c
int main()
{
    return 5 * 8;
}
```

Disassembled code:



```
        .arch armv6
        .eabi_attribute 27, 3
        .eabi_attribute 28, 1
        .fpu vfp
        .eabi_attribute 20, 1
        .eabi_attribute 21, 1
        .eabi_attribute 23, 3
        .eabi_attribute 24, 1
        .eabi_attribute 25, 1
        .eabi_attribute 26, 2
        .eabi_attribute 30, 6
        .eabi_attribute 18, 4
        .file   "2.c"
        .text
        .align  2
        .global main
        .type   main, %function
main:
        @ args = 0, pretend = 0, frame = 0
        @ frame_needed = 1, uses_anonymous_args = 0
        @ link register save eliminated.
        str     fp, [sp, #-4]!
        add     fp, sp, #0
        mov     r3, #40
        mov     r0, r3
        add     sp, fp, #0
        ldmfd   sp!, {fp}
        bx      lr
        .size   main, .-main
        .ident  "GCC: (Debian 4.6.3-8+rpi1) 4.6.3"
        .section        .note.GNU-stack,"",%progbits
```

Last question:

Implementing stack with linked list:

```
#include <stdio.h>
#include <stdlib.h>

typedef struct Node
{
    int val;
    struct Node *next;

}Node;
Node* head;

int pop()
{
    if(head->next == NULL)
    {
        printf("can't pop\n");
        return 0;
    }

    Node* next = head->next->next;
    int val = head->next->val;
    free(head->next);
    head->next = next;
    return val;
}
int top()
{
    if(head->next == NULL)
    {
        printf("stack is empty\n");
        return 0;
    }
    printf("The top value is: %d\n",head->next->val);
    return 1;
}
int push(int val)
{
    Node* node = (Node*)malloc(sizeof(Node));
    node->val = val;
    Node* next = head->next;
    head->next = node;
```

```
    node->next = next;
    return 1;
}
int display()
{
    if(head->next == NULL)
    {
        printf("stack is empty\n");
        return 0;
    }
    for(Node* cur = head->next; cur != NULL; cur = cur->next)
    {
        printf("%d ", cur->val);
    }
    printf("\n");
    return 1;
}
```

```
59  int main(void)
60  {
61      head = (Node*)malloc(sizeof(Node));
62      display();
63      push(1);
64      display();
65      push(2);
66      display();
67      push(3);
68      display();
69      top();
70      display();
71      pop();
72      display();
73      pop();
74      display();
75      pop();
76      display();
77      pop();
78      return 0;
79  }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
(base) alireza@alirezas-MacBook-Air 4 % gcc 4.c
(base) alireza@alirezas-MacBook-Air 4 % ./a.out
stack is empty
1
2 1
3 2 1
The top value is: 3
3 2 1
2 1
1
stack is empty
can't pop
(base) alireza@alirezas-MacBook-Air 4 %
```