CSE 107 LAB 7 2nd SESSION

November 14, 2024

Question 1 (25 points): Given the following C code, what is the expected output when it is executed?

```c
#include <stdio.h>

int main() {
    int n = 5;
    int i, j;

    for (i = 1; i <= n; i++) {
        for (j = 1; j <= n - i; j++) {
            printf("-");
        }
        for (j = 1; j <= 2 * i - 1; j++) {
            printf("*");
        }
        printf("\n");
    }


    for (i = n - 1; i >= 1; i--) {
        for (j = 1; j <= n - i; j++) {
            printf("-");
        }
        for (j = 1; j <= 2 * i - 1; j++) {
            printf("*");
        }
        printf("\n");
    }

    return 0;
}
```

Answer:

Question 2 (25 points): Consider the following C code that is intended to reverse the digits of a given integer `num`:

```c
#include <stdio.h>

int main() {
    int num = 1234;
    int reversed = 0;

    // Attempt to reverse the number
    while (num > 0) {
        reversed = num % 10;
        num = num / 10;
    }

    printf("The reversed number is: %d\n", reversed);

    return 0;
}
```

1. **What is the expected output of the code above?**

2. **Does the algorithm correctly reverse the number? If not, what is the issue with the implementation?**

3. **How should the code be modified to correctly reverse the number?**

    - Describe or write the corrected version of the number reversal algorithm.

Question 3 (25 points): Translate the following C code into assembly-like instructions.

C Code

```c
#include <stdio.h>

int main() {
    int x = 7;
    int y = 2;
    int result = 0;
    int counter = 0;

    while (counter < 4) {
        result = result + x;
        y = y + 1;
        counter++;
    }

    printf("Result: %d, Final y: %d\n", result, y);
    return 0;
}
```

The expected output is "Result: 28, Final y: 6". Write an equivalent assembly-like code that performs the same operations as the C code above. Use the given manual page for opcodes.

Question 4 (25 points): Given the following assembly-like code and the new opcodes added to the manual page, translate the algorithm into C code.

```
load R0, 0
load R5, 5
load R1, 8
load R2, 1
load R4, 3
load R3, 5
addi R3, R3, R2
sub R1, R1, R4
sub R5, R5, R2
jmpLE R5,11
jmp 6
Halt
```

New Opcodes Description

- `sub RST`: Subtract data from register `S` and register `T`, saving the result to register `R`.
- `jmpLE RXY`: Jump to the instruction in memory cell with the address `XY`, if the data in register `R` is less than or equal to the data in register `0`.

Translate the given assembly-like code into an equivalent C code.