- CPSC 275: Introduction to Computer Systems

CPSC 275: Introduction to Computer Systems

Fall 2025

- Syllabus
- Schedule
- Resources
- Upload
- Solution

Protected: Lab 13: Graded

> **IMPORTANT!** This laboratory is a graded, individual assignment. You may use the lecture slides, your own prior laboratory work, and your previous assignments, but you may not use any other electronic resources or the work of other students. Only the computers in the laboratory may be used.

Exercise 1 (50 points)

Write an IA-32 assembly function called `reverse` in a file named `reverse.s` that reverses an array of integers. Your function should accept the array and the number of integers in the array as parameters. You are not allowed to allocate another array; the reversal must be done *in place*. To test your function, write a C program named `reverse.c` that:

1. Allocates an array of integers.
2. Reads a sequence of integers from standard input into the array.
3. Reverses the array by calling your `reverse` function written in assembly.
4. Prints the reversed array.

Compile your program with:

```
$ gcc -m32 -o reverse reverse.s reverse.c
```

Run your program with:

```
$ ./reverse < reverse.in
```

where `reverse.in` is an input file containing the integers for the array. The maximum number of integers is 50.

Sample Input

```
42 139 111 87 40 199 48 134 60 128
```

Output

```
128 60 134 48 199 40 87 111 139 42
```

Programming Notes

- You may use **only** the IA-32 instructions and directives listed here, as well as those covered in lectures and laboratories.
- Use `gdb` to debug your programs, as it will make the process much easier.
- Your program must have the following structure:

```
# program header comment
#
-- definitions such as strings
   .globl main
main:
   pushl %ebp
   movl  %esp,%ebp
   .
   .
   .
   leave
   ret

# function header comment
#
func1:
   .
   .
```

Documentation and Style

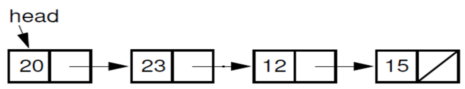
Your program should start with a header comment that includes its name, author, purpose, and last modified date. The code section must be structured into four columns separated by tabs: labels, opcodes, operands, and comments. Be sure to annotate your code to explain the function of each instruction.

Exercise 2 (50 points)

Consider the following declaration for a single node in the linked list of integers:

```
struct Node {
    int data;
    struct Node *next; // Pointer to the next node in the list
};
```

For example, the following linked list contains 4 integers chained together, referenced by the pointer `head`:



Implement the following functions for the linked list:

```
/**
 * Creates a new linked list node with 'data' using malloc and
 * returns a pointer to the newly allocated node, or NULL if
 * malloc failed.
 */
struct Node *createNode(int data);

/**
 * Inserts a new node with 'data' at the head of the list
 */
void insertNode(struct Node *head, int data);

/**
 * Prints all elements in the linked list pointed by 'head'
 */
void printList(struct Node *head);

/**
 * Removes the node at the head of the list
 */
void removeNode(struct Node *head);

/**
 * Frees all memory allocated for the linked list pointed by 'head'
 */
void freeList(struct Node *head);
```

Write the `main()` function to:

1. Define the `head` pointer.
2. Read integers from standard input and insert them at the head of the list.
3. Print the integers in the list by traversing the list.
4. Dispose the list by removing all integers in the list.

Compile your program with:

```
$ gcc -o llist llist.c
```

Run your program with:

```
$ ./llist < llist.in
```

where `llist.in` is an input file containing the integers for the array.

Handin

- **Important!** ***A program with compilation errors will receive a zero***. Make sure your program compiles successfully and runs on our lab computers. Also, eliminate all compiler warnings, or points will deducted.
- Upload your source (`reverse.s`, `reverse.c`, `llist.c`) to the course website **by 4:10 p.m.**

- **Welcome: Sean**

- LogOut