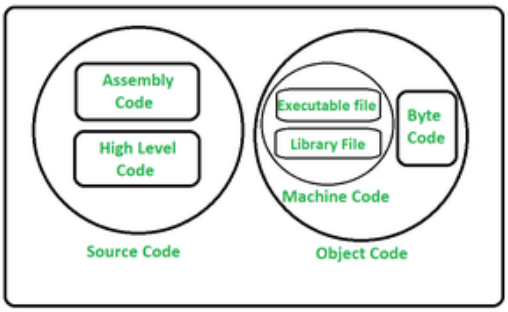
CS 201: Computer Systems

Assignment 2: Object Code



Academic Integrity

You may NOT, under any circumstances, begin a programming assignment by looking for completed code on StackOverflow or Chegg or any such website, which you can claim as your own. Please check out the Student Code of Conduct at PCC.

The only way to learn to code is to do it yourself. The assignments will be built from examples during the lectures, so ask for clarification during class if something seems confusing. If you start with code from another source and just change the variable names or other content to make it look original, you will receive a zero on the assignment.

I may ask you to explain your assignment verbally. If you cannot satisfactorily explain what your code does, and answer questions about why you wrote it in a particular way, then you should also expect a zero.

Purpose

The purpose of this assignment is to be able to write code with the common arithmetic instructions used to perform integer math.

After completing this assignment you will be able to:

- Explain how to obtain assembly code from source code or from a program.
- Understand the general environment that assembly code works in: registers, program stacks, etc.
- Use the most common assembly code instruction: the mov instruction.
- Write code with the common arithmetic instructions used to perform integer math.

Task

- Download the `gccinline.c` file.
- There are two functions you need to fill in. In the function regular_pay_asm(), write the assembly code needed to calculate regular pay.
- Regular pay is defined as pay rate multiplied by the number of hours equal to or under 40. So the algorithm would be:

```
if (hours > 40) hours = 40
pay = hours x payrate
```

- In the function regular_pay_C() write the C equivalent to the above algorithm in C code.
- This should be written first so you can compare the output of the asm code to the C code.
- Do not write any C code within the regular_pay_asm() function; only write assembly code to do the task. I have provided you with the necessary framework so all that is needed is to fill in the assembly code.

Criteria for Success

- For information on how to do this assignment, see week 4 and week 5 lecture pages on Object Code in the D2L shell.
- Test your program using any given sample runs or your own data multiple times.
- Upload the C source code file, with your assembly code (using extended gcc inline assembly) per the instructions in the D2L Assignment folder for assignment 2 by the due date. ***Some instructors require you to turn in your assignment on D2L while some others like it turned into the Linux server. Be sure to read the instructions in the assignment folder.***
- Do your own work. Consult the syllabus for more information about academic integrity.
- NOTE: see the GCC-Inline-Assembly website for more information on how to do gcc inline assembly.

Sample Runs

```
----------------------
EXAMPLE RUN #1:
----------------------
enter pay rate: 21
enter hours: 30
regular pay (C) is: 630
regular pay (asm) is: 630
```

```
----------------------
EXAMPLE RUN #2:
----------------------
enter pay rate: 21
```

```
enter hours: 54
regular pay (C) is: 840
regular pay (asm) is: 840
```

```
----------------------
EXAMPLE RUN #3:
----------------------
enter pay rate: 21
enter hours: 40
regular pay (C) is: 840
regular pay (asm) is: 840
----------------------
```

Grading Criteria

- The modified code compiles with gcc without errors.
- The program does not have run-time errors.
- The modified code produces accurate results.
- Both given functions are filled in.
- The function regular_pay_asm() does all calculations using only gcc inline assembly code.
- The function regular_pay_C() is all C code and correctly calculates the regular pay.
- The two functions always match for all possible valid values.
  NOTE: Valid values for both pay rate and hours are integers bigger than zero (but not zero).