Refresher Module Assignment 0.1

Report

The Makefile contains a total of 6 flags:

```
1  default:
2      gcc hello.c -o hello
3      ./hello
4  preprocess:
5      gcc -E hello.c -o hello.i
6  compile:
7      gcc -S hello.i -o hello.s
8  assemble:
9      gcc -c hello.s -o hello.o
10 link:
11     gcc hello.o -o hello
12 run:
13     ./hello
```

1. **default**
   - This flag will run by default in the makefile if no other flag is specified.
   - This flag will compile the c program (hello.c) and run the program.
   - The codes used under this flag is:

     gcc hello.c -o hello

     ./hello

```
lalitwadhwa@ubuntu:~/Desktop/Sem3/OS/Assignment0/0.1$ make
gcc hello.c -o hello
./hello
Value of var1 = 10
Value of var2 = 20
```

   - The first command above will convert the hello.c file to an executable file.

- The name of the executable file would be hello courtesy of -o command line option
- The second command will run the hello file created in the first step.

2. **preprocess**
   - This flag will preprocess the file (hello.c) and convert it to .i format
   - To call this flag user would have to type:

     make preprocess
   - The code used under this flag is:

     gcc -E hello.c -o hello.i

```
lalitwadhwa@ubuntu:~/Desktop/Sem3/OS/Assignment0/0.1$ make preprocess
gcc -E hello.c -o hello.i
```

   - The newly created file would be named as hello.i courtesy -o command line option.
   - The -E is one of the overall options used in gcc.
   - This option stops the process at the preprocessing stage.
   - The file obtained at this step, hello.i, will have the following changes to it:
     - The header files mentioned in the actual c program would now be expanded and included in the .i file.
     - The macros would be expanded in the program.
     - The comments, if any, in the c code would be removed in the .i file.

```
# 1 "hello.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 31 "<command-line>"
# 1 "/usr/include/stdc-predef.h" 1 3 4
# 32 "<command-line>" 2
# 1 "hello.c"
# 1 "/usr/include/stdio.h" 1 3 4
# 27 "/usr/include/stdio.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/libc-header-start.h" 1 3 4
# 33 "/usr/include/x86_64-linux-gnu/bits/libc-header-start.h" 3 4
# 1 "/usr/include/features.h" 1 3 4
# 424 "/usr/include/features.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 1 3 4
# 427 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
# 428 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 2 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/long-double.h" 1 3 4
# 429 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 2 3 4
# 425 "/usr/include/features.h" 2 3 4
```

3. **compile**
   - This flag will compile the file (hello.i) and convert it to .s format
   - To call this flag user would have to type:

     make compile
   - The code used under this flag is:

     gcc -S hello.i -o hello.s

```
lalitwadhwa@ubuntu:~/Desktop/Sem3/OS/Assignment0/0.1$ make compile
gcc -S hello.i -o hello.s
```

   - The newly created file would be named as hello.s courtesy -o command line option.
   - The -S is one of the overall options used in gcc.
   - This option stops the process at the compilation stage.
   - The file obtained at this step, hello.s, will have the following changes to it:
     - It would be in assembly code
     - The contents of the .i file would be exactly written in the assembly code supported by the architecture of the machine the user would be using

```
1     .file   "hello.c"
2     .text
3     .section    .rodata
4 .LC0:
5     .string "Value of var1 = %d\n"
6     .text
7     .globl  main
8     .type   main, @function
9 main:
10 .LFB0:
11    .cfi_startproc
12    pushq   %rbp
13    .cfi_def_cfa_offset 16
14    .cfi_offset 6, -16
15    movq    %rsp, %rbp
16    .cfi_def_cfa_register 6
17    subq    $16, %rsp
18    movl    $10, -8(%rbp)
19    movl    $20, -4(%rbp)
20    movl    -8(%rbp), %eax
```

4. **assemble**
   - This flag will assemble the file (hello.s) and convert it to .o format
   - To call this flag user would have to type:

     make assemble
   - The code used under this flag is:

     gcc -c hello.s -o hello.o

```
lalitwadhwa@ubuntu:~/Desktop/Sem3/OS/Assignment0/0.1$ make assemble
gcc -c hello.s -o hello.o
```

   - The newly created file would be named as hello.o courtesy -o command line option.
   - The -c is one of the overall options used in gcc.
   - This option stops the process at the assembly stage.
   - The file obtained at this step, hello.o, will have the following changes to it:
     - It would be in machine code (binary/hexadecimal) as supported for viewing in your machine
     - The contents of the .o file would be exactly written in binary as they were written in the assembly code.

```
000000b0  2d 33 75 62 75 6e 74 75  31 7e 31 38 2e 30 34 29
000000c0  20 37 2e 35 2e 30 00 00  14 00 00 00 00 00 00 00
000000d0  01 7a 52 00 01 78 10 01  1b 0c 07 08 90 01 00 00
000000e0  1c 00 00 00 1c 00 00 00  00 00 00 00 49 00 00 00
000000f0  00 41 0e 10 86 02 43 0d  06 02 44 0c 07 08 00 00
00000100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000110  00 00 00 00 00 00 00 00  01 00 00 00 04 00 f1 ff
00000120  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000130  00 00 00 00 03 00 01 00  00 00 00 00 00 00 00 00
00000140  00 00 00 00 00 00 00 00  00 00 00 00 03 00 03 00
00000150  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000160  00 00 00 00 03 00 04 00  00 00 00 00 00 00 00 00
00000170  00 00 00 00 00 00 00 00  00 00 00 00 03 00 05 00
```

5. **link**
   - This flag will link the file (hello.o) and convert it to executable format
   - It could also have been converted to .out format as that can also be directly used to run the final code.
   - To call this flag user would have to type:

     make link

- The code used under this flag is:

  gcc hello.o -o hello

```
lalitwadhwa@ubuntu:~/Desktop/Sem3/OS/Assignment0/0.1$ make link
gcc hello.o -o hello
```

- The newly created executable file would be named as hello courtesy -o command line option.
- The file obtained at this step, hello, will have the following changes to it:
  - It would be an executable file
  - The file can run by directly using ./hello command

6. **run**
   - This flag will run the executable file hello
   - To call this flag user would have to type:

     make run
   - The code used under this flag is:

     ./hello

```
lalitwadhwa@ubuntu:~/Desktop/Sem3/OS/Assignment0/0.1$ make run
./hello
Value of var1 = 10
Value of var2 = 20
```

WARNING: For commands from number 2 to number 6, since it was given in the question that we would have to pause at each step, therefore the file generated in the previous step is used to generate a new file. If this order is not followed, then an error would be generated by the command line. However this is not true for the default flag (command 1).