1. The -S command tells the compiler to stop at the assembly level. The -O0 disables optimization
2. The .s is the compiled assembly code for the .c file
3. This command assembles the assembly source into an object .o file which can be turned into an executable
4. Makes an executable out of the object file
5. No output is specified in the code
   1. Set a breakpoint at line 10 (Or first executable line after 10) effect: adds breakpoint at line 26
   2. Run code until breakpoint
   3. Show registers in terminal
   4. Step through a single line of code
   5. Show state of registers
   6. Add a breakpoint at line 12 (or first executable line after 10) effect: adds breakpoint at line 26
   7. Continue until next break (ends the code since no more breakpoints
   8. Lists current breakpoints
   9. Deletes first breakpoint
   10. Lists breakpoints
   11. Deletes second breakpoint
   12. Continue (Nothing happens since program has already exited)
   13. quit
6. Assembles the assembly source into an object file
7. Makes an executable out of the object file
   1. Sets breakpoint at line 19
   2. Runs code until breakpoint
   3. Show registers
   4. Steps two lines down
   5. Show registers
   6. Set breakpoint at line 24
   7. Continue until breakpoint
   8. Show breakpoints
   9. Delete b1
   10. Show breakpoints
   11. Delete b2
   12. Show breakpoints
   13. Finish program
   14. Quit
8. Assembles the .s file to an object file. Stores a detailed listing of the assembler output to the file helloworld.lst
9. 1: memory location of the instruction 2: binary instruction code 3: corresponding assembly code
10. Gdb
    1. Set a breakpoint at line 26
    2. Run until breakpoint
    3. Show address stored in helloMsgAddr
    4. Show data stored at memory location 0x000104ac (decimal and ascii)
    5. Show first 5 bytes of data (decimal and ascii)
    6. Show first 14 bytes of data (hex)
    7. Show first 5 bytes of data (hex)
    8. Show first 5 bytes of data (binary)
    9. Run the code to completion (no more breakpoints)
11. gdb
    1. set breakpoint at line 49
    2. run until bp
    3. show character stored at frame pointer-5 (typed character)
    4. complete program
12. the program loops 5 times, turning the LED on, waiting a second, turning it off, waiting a second, and decrementing the counter.

sub sp, sp, 8 @ space for fp, lr

str fp, [sp, 0] @ save fp

str lr, [sp, 4] @ and lr

add fp, sp, 4 @ set our frame pointer

sub sp, sp, local @ allocate memory for local var

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | init |  |  | sub sp, sp, 8 | |  | str lr, [sp, 4] | |
|  |  |  |  |  |  |  | add fp, sp, 4 | |
|  |  |  |  |  |  |  |  |  |
|  |  |  | > | 0xbefff1b0 | ##### | > | 0xbefff1b0 | 0x0 |
|  |  |  |  | 0xbefff1b4 | ##### |  | 0xbefff1b4 | 0xb6e6c718 |
| > | 0xbefff1b8 | ####### |  | 0xbefff1b8 | ##### |  | 0xbefff1b8 | ##### |
|  | sub sp, sp, local | |  | add sp, sp, local | |  | add sp, sp, 8 | |
| > | 0xbefff1a8 | ##### |  | 0xbefff1a8 | ##### |  | 0xbefff1a8 | ##### |
|  | 0xbefff1ac | ##### |  | 0xbefff1ac | ##### |  | 0xbefff1ac | ##### |
|  | 0xbefff1b0 | 0x0 | > | 0xbefff1b0 | 0x0 |  | 0xbefff1b0 | 0x0 |
|  | 0xbefff1b4 | 0xb6e6c718 |  | 0xbefff1b4 | 0xb6e6c718 |  | 0xbefff1b4 | 0xb6e6c718 |
|  | 0xbefff1b8 | ##### |  | 0xbefff1b8 | ##### | > | 0xbefff1b8 | ##### |