Assembly programming

.global main

* Necessary to run program

gdb

* GNU Debugger
  + Text command line interface
* Sp – stack pointer
* Lr - link regester
* Fp – frame pointer
* Pc – Program counter

Commands

* disassemble main
  + dumps assembly code for function
  + Outputs it as hex addresses and the code
* lr
  + link regester
* bl
  + branch on link
    - calls function

Compiling

* Important Assembly rules
  + Put alll variable cosntants and literals into the data section
  + Make all functions and program constructs as label in the .text sectuib
    - Ex) .Text, .global\_main
* Data
  + Contains all literals and values

Example

* .data

//Label

hello\_msg:

//Terminator string returns termination for you

.asciz “ Hello everyone!\n“

fmt\_string:

.asciz “x=%d ......\n“

x: .word 1

y: .word 3

z: .word 5

a: .word 42

b: .word 35

c: .word 21

Issues with Stack

* Printf only has registers from r0-r3
* R4 onward must be pushed
* After the printf, used regiesters must be popped
  + Synatx) pop{r6}
  + Methods)
    - Same register pop
      * (if there are 3 variables registered)
      * Syntax) pop{r4} pop{r4} pop{r4}
    - Top down pop
      * Syntax) pop{r6} pop{r5} pop{}