GNU Assembly Language **Introduction** worksheet:

What is a ***label*** in Assembly Language **and** how is it used?

Go to: https://en.wikipedia.org/wiki/Label_(computer_science)

Suffixes to Commands:

Go to: https://en.wikibooks.org/wiki/X86_Assembly/GAS_Syntax#Operation_Suffixes

suffix:	meaning:
b	
s	
w	
l	
q	
t	

What purpose does the suffix serve?

Are you required to use a suffix?

If you use the command without a suffix what happens?

Commands:

Go to: https://en.wikibooks.org/wiki/X86_Assembly/GAS_Syntax#Quick_reference

and http://cs.neiu.edu/fporps/2021Fall/cs301/CS301-08machine-basics.pdf (note: change 2021Fall to current semester)

instruction with sample	meaning/how it works:
movq %rax, %rbx	
movq $123, %rax	
movq %rsi, -16(%rbp)	
subq $10, %rbp	
cmpl %eax %ebx	
jmp location	
je location	
jg, jge, jl, jle, jne, …	
leaq	
salq	
addq	
subq	
imulq	
sarq	
shrq	
xorq	
andq	
orq	
incq	
decq	
negq	
notq	
*	

*These are most common, but you can search the web for more (not required).

Registers and storage (memory) addressing: (This is your workspace in assembly language)

Go to: https://cs61.seas.harvard.edu/site/2018/Asm1/

Complete the register reference chart

General Purpose Registers:

Special Purpose Registers:

Sample Code To Run (2 programs – copy, paste, and run individually):

exit program to "see" output Note: # ←used to mark a COMMENT (invisible ink) in Assembly Language

```
#PURPOSE:       Simple program that exits and returns a
#                       status code back to the Linux kernel
#
#INPUT:         none
#
#OUTPUT:        returns a status code. This can be viewed
#                       by typing
#
#                       echo $?
#
#                       after running the program
#
#VARIABLES:
#                       %rax holds the system call number
#                       (this is always the case)
#
#                       %rbx holds the return status
#
.section .data

.section .text

.globl _start
_start:
movq $1, %rax      # this is the linux kernel command
                # number (system call) for exiting
                # a program

movq $0, %rbx      # this is the status number we will
                                        # return to the operating system.
                                        # Change this around and it will
                                        # return different things to
                                        # echo $?
int $0x80                       # this wakes up the kernel to run
                                        # the exit command
```

max program This program finds the maximum number of a set of data items.

```
#PURPOSE:  This program finds the maximum number of a
#          set of data items.
#
#VARIABLES: The registers have the following uses:
#
# %rdi - Holds the index of the data item being examined
# %ebx - Largest data item found
# %eax - Current data item
#
# The following memory locations are used:
#
# data_items - contains the item data. A 0 is used
# to terminate the data
#
.section .data

data_items:          #These are the data items
.int 3,67,34,222,45,75,54,34,44,33,22,11,66,0

.section .text
.globl _start
_start:
movq $0, %rdi                    # move 0 into the index register
movl data_items(,%rdi,4), %eax   # load the first byte of data
movl %eax, %ebx                  # since this is the first item, %rax is
                                 # the biggest
start_loop:                      # start loop
cmpl $0, %eax                    # check to see if we've hit the end
je loop_exit
incq %rdi                        # load next value
movl data_items(,%rdi,4), %eax
cmpl %ebx, %eax                  # compare values
jle start_loop                   # jump to loop beginning if the new
                                 # one isn't bigger
movl %eax, %ebx                  # move the value as the largest
jmp start_loop                   # jump to loop beginning
loop_exit:
# %rbx is the return value, and it already has the number
movq $1, %rax                    #1 is the exit() syscall
int $0x80
```