SASM emulator for NASM target

This document provides information on how to use a NASM (Intel Assembler) Emulator named SASM to confirm that your compiler is generating proper NASM code. This document is not exhaustive. There are many resources on NASM via a web search. This document provides details on how to get started with SASM

SASM is an open source emulator of several assembly languages. We will be using the x64 version of NASM for our work (this is a setup option in SASM). This emulator differs from a barebones NASM assembler (found on Windows and linux), in that INPUT and OUTPUT is generic to the platform. This means that if you want to run your code directly on an Intel platform, then you will need to rework the read and write methods.

SASM can be found at https://dman95.github.io/SASM/english.html

NASM is one form of Intel Assembler. You can find details on NASM at https://en.wikipedia.org/wiki/Netwide_Assembler

To simplify the various ways to run NASM on SASM, we need to know/program the following subroutine linkage:

- The "main" function must have first as the first directive: "mov   rbp, rsp"
  - This is required to set the Base Pointer to the Current Stack pointer
  - We need this activity because of the linkage between the simulator and the start of the external "main"
- A "call" to a function decrements the RSP and places the return address on the stack (this is normal)
- You need to make each function's RBP the same as RSP and the start of the function
- You must decrement RSP to leave space to work your activation record
- You must return RSP and RBP back to their original value before you return (except main, where RSP and RBP will be the values of RSP).
- We will use "io64.inc" to deal with our I/O (read and write)
  - you should have the following line in every code head
    %include "io64.inc"

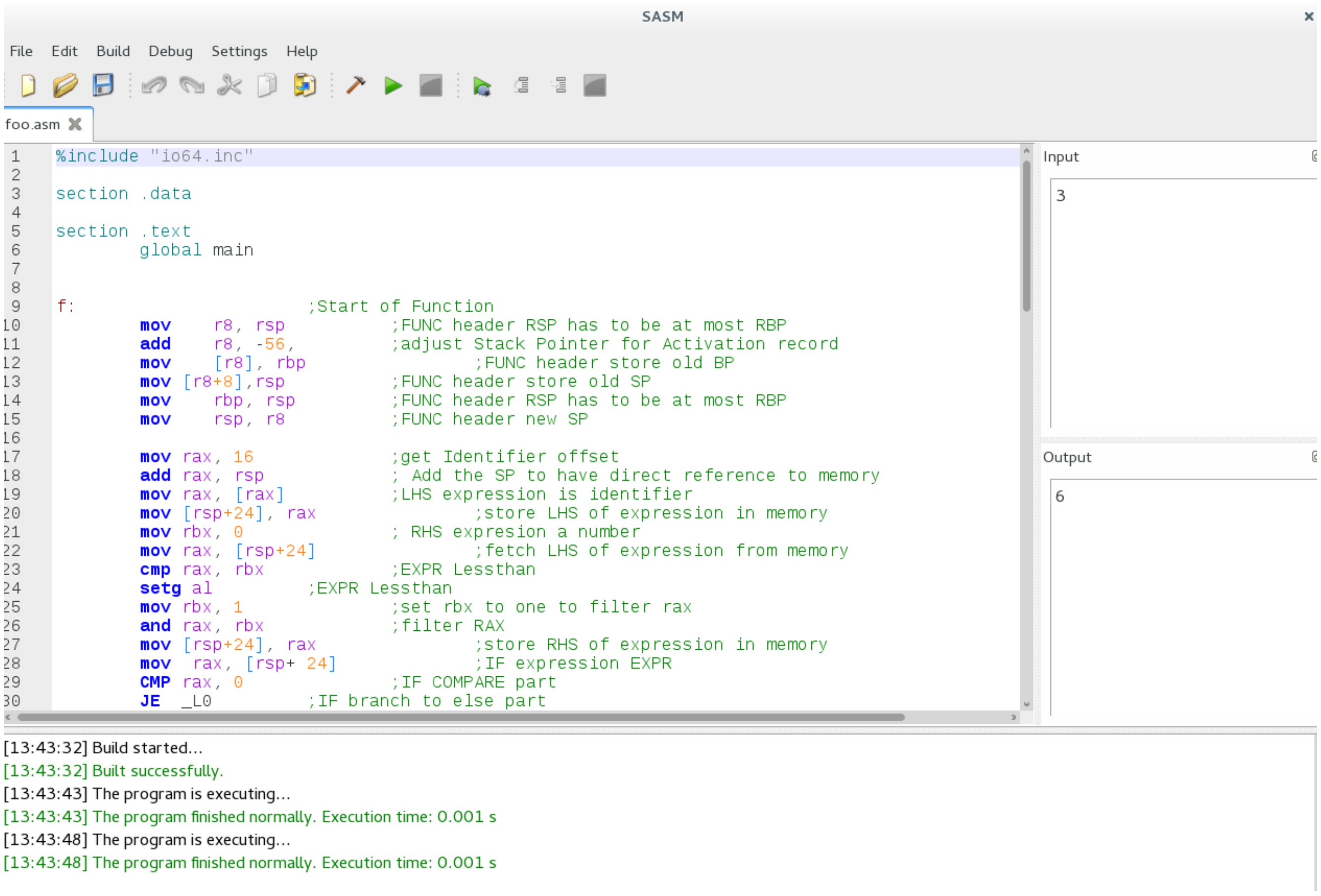
2) We will be working only with 64 bit registers. This means that all words have 8 bytes.

How to use SASM from the CS domain.

Start "sasm" from a terminal

1. Your input file should have the suffix ".asm"
2. Set your settings to x64, NASM (see below for an image)
3. Use the "Build" option to assemble your code and find errors
4. Use the "build and run" option to run your code

5. Use the “Step Into” single stepper for better control of the emulator
6. The assembler is not interactive. For code which requires input, you will need to put your input values in the INPUT window, prior to the start of your program. Y

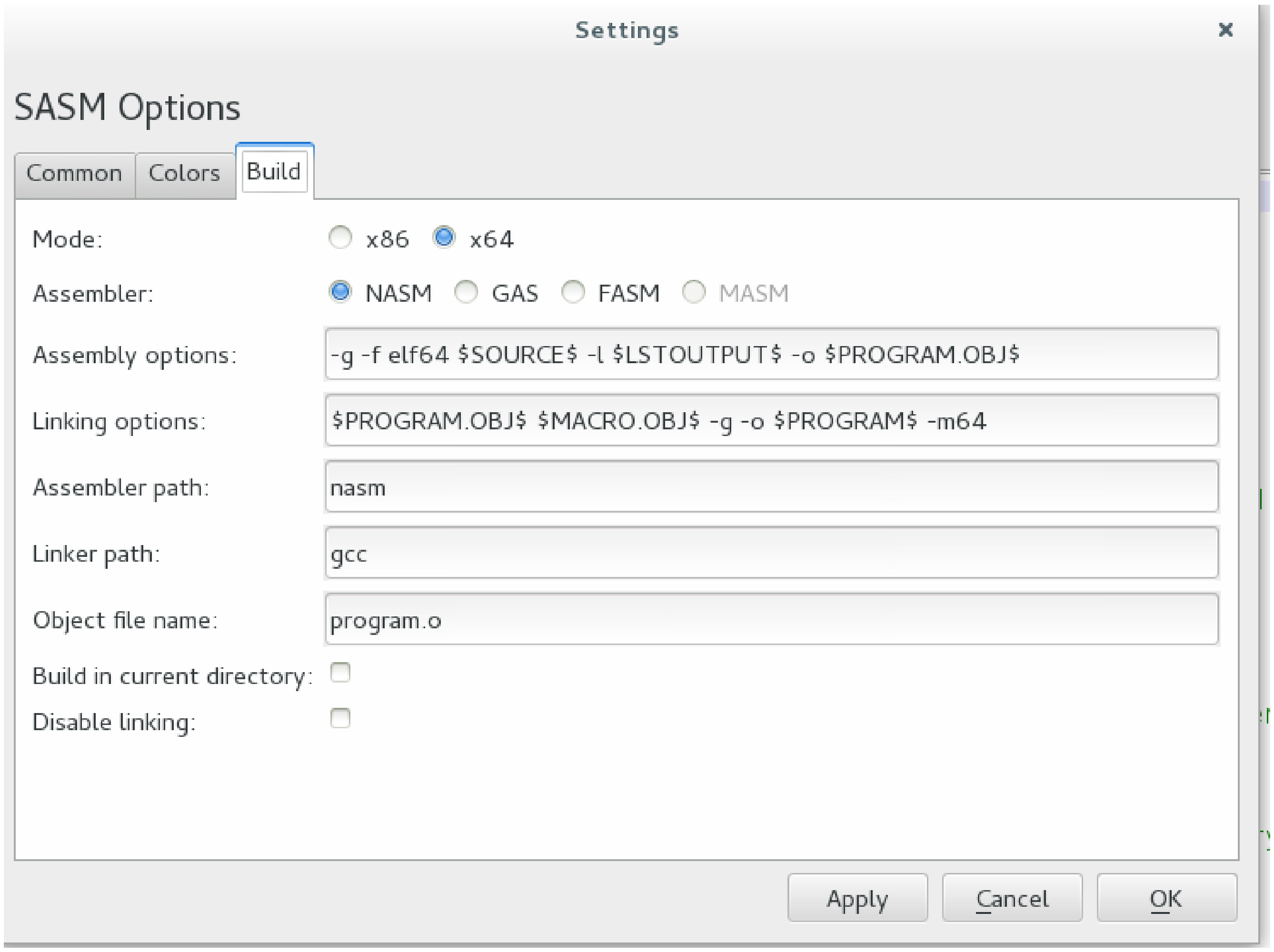


```
%include "io64.inc"

section .data

section .text
        global main


f:                      ;Start of Function
        mov    r8, rsp          ;FUNC header RSP has to be at most RBP
        add    r8, -56,         ;adjust Stack Pointer for Activation record
        mov    [r8], rbp                ;FUNC header store old BP
        mov [r8+8],rsp          ;FUNC header store old SP
        mov    rbp, rsp         ;FUNC header RSP has to be at most RBP
        mov    rsp, r8          ;FUNC header new SP

        mov rax, 16             ;get Identifier offset
        add rax, rsp            ; Add the SP to have direct reference to memory
        mov rax, [rax]          ;LHS expression is identifier
        mov [rsp+24], rax               ;store LHS of expression in memory
        mov rbx, 0              ; RHS expression a number
        mov rax, [rsp+24]               ;fetch LHS of expression from memory
        cmp rax, rbx            ;EXPR Lessthan
        setg al         ;EXPR Lessthan
        mov rbx, 1              ;set rbx to one to filter rax
        and rax, rbx            ;filter RAX
        mov [rsp+24], rax               ;store RHS of expression in memory
        mov  rax, [rsp+ 24]             ;IF expression EXPR
        CMP rax, 0              ;IF COMPARE part
        JE  _L0         ;IF branch to else part
```

```
3
```

```
6
```

```
[13:43:32] Build started...
[13:43:32] Built successfully.
[13:43:43] The program is executing...
[13:43:43] The program finished normally. Execution time: 0.001 s
[13:43:48] The program is executing...
[13:43:48] The program finished normally. Execution time: 0.001 s
```



SASM Options

Mode: x86 [✓ x64]

Assembler: [✓ NASM] GAS FASM MASM

Assembly options: -g -f elf64 $SOURCE$ -l $LSTOUTPUT$ -o $PROGRAM.OBJ$

Linking options: $PROGRAM.OBJ$ $MACRO.OBJ$ -g -o $PROGRAM$ -m64

Assembler path: nasm

Linker path: gcc

Object file name: program.o

Build in current directory: ☐

Disable linking: ☐