```
            section .data

L1:         db      'PUSH BYTE   : '                        ;Labels
L2:         db      'PUSH 16BIT  : '
L3:         db      'PUSH 64BIT  : '
L4:         db      'PUSH FLOAT  : '

OUTPUT:     db      '                                   '    ;
HEX:        db      '0123456789ABCDEF'                      ;hex table
LF:         db      0x0a                                    ;line feeds
LF2:        db      0x0a,0x0a

SPTRB:      dd      -1                                      ;Stack Pointer offsets
SPTR16:     dd      -1
SPTR64:     dd      -1
SPTRF:      dd      -1


f1:         dw 1.1                                          ;sample float value

            section .bss
STACKB:     resb 32                                         ;Stacks
STACK16:    resw 32
STACK64:    resq 32
STACKF:     resd 32




            section .text
            global  main                                    ;Tell linker about main
            extern  write, exit
main:
            mov     rbp, rsp                                ; for correct debugging
            push    rbp
            mov     rbp, rsp


            lea     rsi,[L1]
            call    WRITELBL
            mov     al,0xFF                                 ;value to push
            call    PUSHB
            mov     al,0                                    ;clear al
            call    POPB                                    ;pop, al s/b 0xFF
            mov     rcx,2
            call    TOHEX
            mov     edx,2
            call    MYWRITE
            call    WRITELF

            lea     rsi,[L2]
            call    WRITELBL
            mov     ax,0xFFFF                               ;value to push
            call    PUSH16
            mov     ax,0x0                                  ;clear ax
            call    POP16                                   ;pop, ax s/b 0xFFFF
            mov     rcx,4
            call    TOHEX
            mov     edx,4
            call    MYWRITE
            call    WRITELF

            lea     rsi,[L3]
            call    WRITELBL
            mov     rax,0xFFFFFFFFFFFFFFFF                  ;value to push
            call    PUSH64
            mov     rax,0                                   ;clear rax
```

```
            call    POP64                                   ;pop, rax s/b 0xFFFFFFFF
            mov     rcx,16
            call    TOHEX
            mov     edx,16
            call    MYWRITE
            call    WRITELF

            lea     rsi,[L4]
            call    WRITELBL
            mov     eax,[f1]                                ;value to push (single prec 1.1)
            call    PUSHFL
            mov     eax,0                                   ;clear eax
            call    POPFL                                   ;pop rax s/b float 1.1
            mov     rcx,8
            call    TOHEX
            mov     edx,8
            call    MYWRITE
            call    WRITELF



MX:         xor     edi, edi                                ; 0 return = success
            call    exit



;BYTE Operators ------------------------------------------------

;Usage: Put value in AL.
PUSHB:      xor     edx,edx
            mov     edx,[SPTRB]                             ;load stack ptr offset
            cmp     edx,32                                  ; range check
            je      PUSHBX
            inc     edx                                     ;inc stack
            mov     [SPTRB],edx                             ;save stack ptr offset
PUSHBX:     mov     [edx+STACKB],al                         ;push value
            ret

;Usage: Returns value in AL.
POPB:       xor     edx,edx
            mov     edx,[SPTRB]                             ;load stack ptr offset
            mov     al,[edx+STACKB]                         ;get value
            cmp     edx,0                                   ;range check
            je      POPBX
            dec     edx                                     ;dec stack ptr
            mov     [SPTRB],edx                             ;save stack ptr offset
POPBX:      ret


;16bit Operators -----------------------------------------------

;Usage: Put value in ax.
PUSH16:     xor     rdx,rdx
            mov     edx,[SPTR16]                            ;load stack ptr offset
            cmp     edx,32                                  ; range check
            je      PUSH16X
            inc     edx                                     ;inc stack
            mov     [SPTR16],edx                            ;save stack ptr offset
PUSH16X:    mov     [(edx*2)+STACK16],ax                    ;push value
            ret

;Usage: Returns value in AL.
POP16:      xor     rdx,rdx
            mov     edx,[SPTR16]                            ;load stack ptr offset
            mov     ax,[(edx*2)+STACK16]                    ;get value
            cmp     edx,0                                   ;range check
```

```
            je      POP16X
            dec     edx                                 ;dec stack ptr
            mov     [SPTR16],edx                        ;save stack ptr offset
POP16X:     ret




;64bit Operators ------------------------------------------------

;Usage: Put value in rax.
PUSH64:     mov     edx,[SPTR64]                        ;load stack ptr offset
            cmp     edx,32                              ; range check
            je      PUSH64X
            inc     edx                                 ;inc stack
            mov     [SPTR64],edx                        ;save stack ptr offset
PUSH64X:    mov     [(edx*8)+STACK64],rax               ;push value
            ret

;Usage: Returns value in AL.
POP64:      mov     edx,[SPTR64]                        ;load stack ptr offset
            mov     rax,[(edx*8)+STACK64]               ;get value
            cmp     edx,0                               ;range check
            je      POP64X
            dec     edx                                 ;dec stack ptr
            mov     [SPTR64],edx                        ;save stack ptr offset
POP64X:     ret



;FLOAT Operators ------------------------------------------------

;Usage: Put value in eax.
PUSHFL:     xor     rdx,rdx
            mov     edx,[SPTRF]                         ;load stack ptr offset
            cmp     edx,32                              ; range check
            je      PUSHFLX
            inc     edx                                 ;inc stack
            mov     [SPTRF],edx                         ;save stack ptr offset
PUSHFLX:    mov     [(edx*4)+STACKF],eax                ;push value
            ret

;Usage: Returns value in AL.
POPFL:      mov     edx,[SPTRF]                         ;load stack ptr offset
            mov     eax,[(edx*4)+STACKF]                ;get value
            cmp     edx,0                               ;range check
            je      POPFLX
            dec     edx                                 ;dec stack ptr
            mov     [SPTRF],edx                         ;save stack ptr offset
POPFLX:     ret




; Usage: Load RSI with label
WRITELBL:
            mov     edx, 14                             ; write label
            mov     edi, 1
            call    write
            ret

; Usage: CALL
WRITELF:
            lea     rsi,[LF]
            mov     edx, 1                              ; write label
            mov     edi, 1
            call    write
```

```
            ret

; Usage: Load edx with length
MYWRITE:
            lea     rsi,[OUTPUT]
            mov     edi, 1
            call    write
            ret


; Usage: Load rax with value and rcx with length
TOHEX:
            push    qword rbx
            mov     rbx,rax
            lea     edx,[OUTPUT+(rcx-1)]                ;point to end of output string needed
                                                        ;mov rcx,2

TH1:        mov     rax,rbx                             ;loop start
            and     rax,0xF                             ;and to get lowest byte value...
            mov     al,[HEX+eax]
            mov     [edx],al                            ;store number...
            shr     rbx,4                               ;shift working value right for next byte
            dec     edx
            loop    TH1
                                                        ;mov eax,0x20
                                                        ;mov [OUTPUT+3],al
            pop     qword rbx
            ret
```