Running an Assembler program from BASIC using SYS

A Basic program can call Assembler code using the SYS command. See the description of the SYS Basic command. Before calling the specified address, SYS “loads” the accumulator, the X and the Y index register, and the status register with the bytes stored at addresses 780–783/$030C–$030F: From BASIC, one can set up parameters and data here, to be processed by the machine language routine.

This typically leads to games using a Basic stub to call the game assembler code. This leads to a typical line like this:

```
1984 SYS 2084
```

The hexdump shows:

```
>C:0800  00 0c 08 c0  07 9e 20 32  30 38 34 00  00 00 00 00   ......
2084.....
```

At $0801 is the pointer to the next Basic line the following two byte are the little endian number $07c0 which is the line number 1984 the follows $9e which is the Basic token for SYS. After that we have a space and the number 2084 in PETSCII.

A simple stub for use

A simple stub program looks like this (this uses xa65 syntax, see the xa65 homepage). The code:

```
    .word $0801
    * = $0801
basic:  .(
    .word end_of_basic
    .word 2017
    .byte $9e,$20
    .asc "(2064)"
end_of_basic:
    .byte 0,0,0
    .)

main:   sta $0400
    stx $0401
    sty $0402
    php
    pla
    sta $0403
    lda #$ff
    tax
    tay
    rts
```

compiles to:

```
>C:0800  00 0e 08 e1  07 9e 20 28  32 30 36 34  29 00 00 00   ......
(2064)...
>C:0810  8d 00 04 8e  01 04 8c 02  04 08 68 8d  03 04 a9 ff
..........h.....
>C:0820  aa a8 60 00  00 00 00 00  00 00 00 00  00 00 00 00
..`.............
```

As can be easily seen the SYS to address 2064 starts the code at $0810.

Fancier ways to call the Assembler code

Using the current line number

This is the canonical way to call an Assembler code. Usually you precalculate the entry or you have to be very careful when changing the basic stub. Another nice way to call the assembler code is the following stub:

```
    .word $0801
    * = $0801
basic:  .(
    .word end_of_basic
    .word main
    .byte $9e,$20,$c2
    .asc "(57)",$aa,"256",$ac,$c2,"(58)"
end_of_basic:
    .byte 0,0,0
    .)

main:   lda #$0d
    sta $d020
        rts
```

This code uses the zeropage locations $39/$3A which contain the current line number in the basic program, see zeropage description. The line number was selected to be the entry of the main routine.

Using ASC()

If you like to get your code fancier you can use something like this:

```
    .word $0801
    * = $0801
basic:  .(
    .word L1
    .word main
    .byte $9e
```

```
    .asc "(",$c6,"(",$22
    .byte ((main>>6)&$3f)+$20
    .asc $22,")",$ab,"32)",$ac,"64",$aa,$c6,"(",$22
    .byte ((main)&$3f)+$20
    .asc $22,")",$ab,"32"
    .asc 0
L1: .word end_of_basic
    .word L1
    .byte $8f,$20
    .asc "ANOTHER VISITOR STAY A WHILE...",0
end_of_basic:
    .word 0,0
    .)

main:   lda #$0d
    sta $d020
        rts
```

This use two characters to encode the starting address (maximum is 4096). The basic code actually reads:

```
2121 SYS(ASC("A")-32)*64+ASC(")")-32
2079 REM ANOTHER VISITOR STAY A WHILE...
```

The line numbers are even not ordered but this code safely runs.