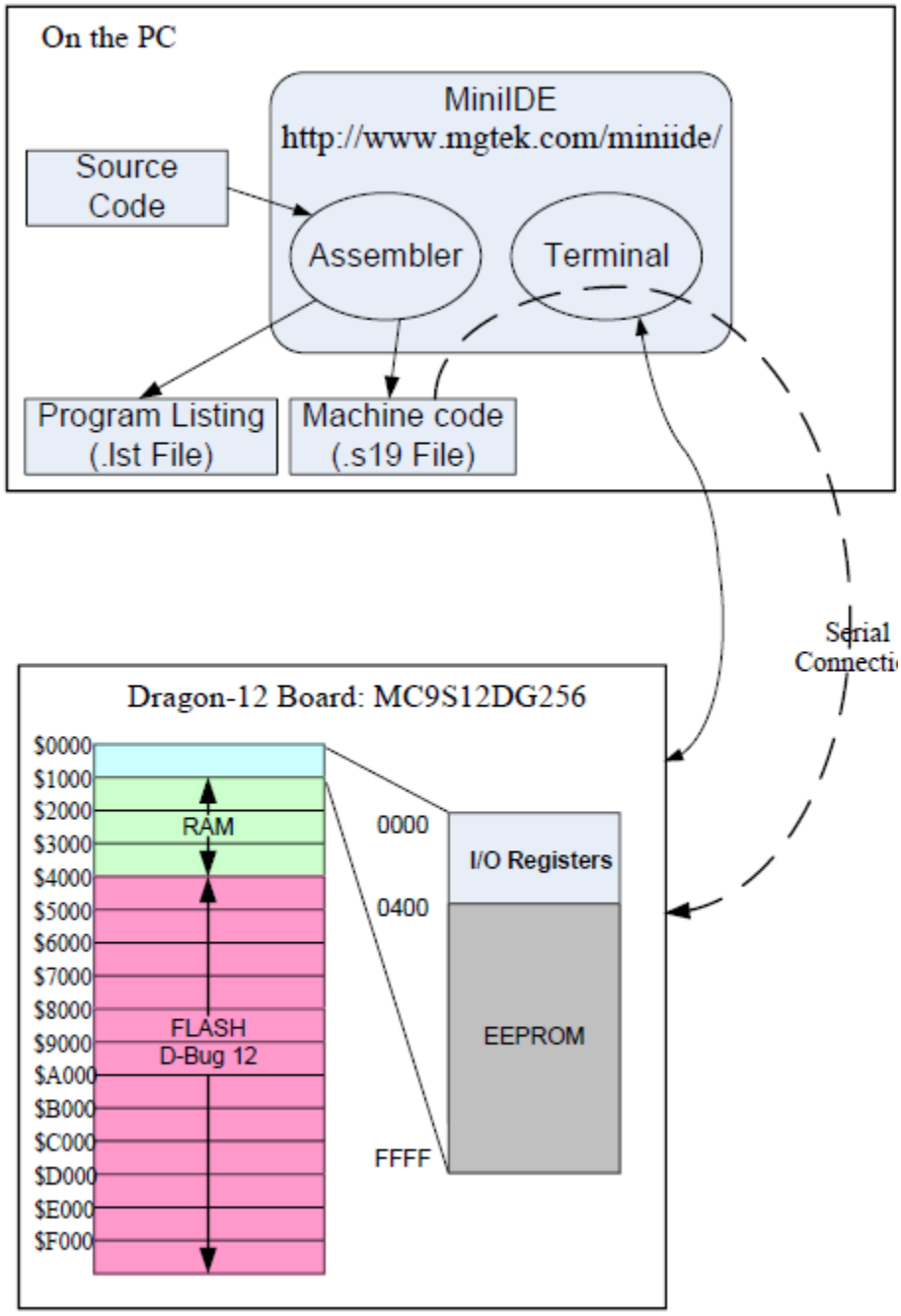
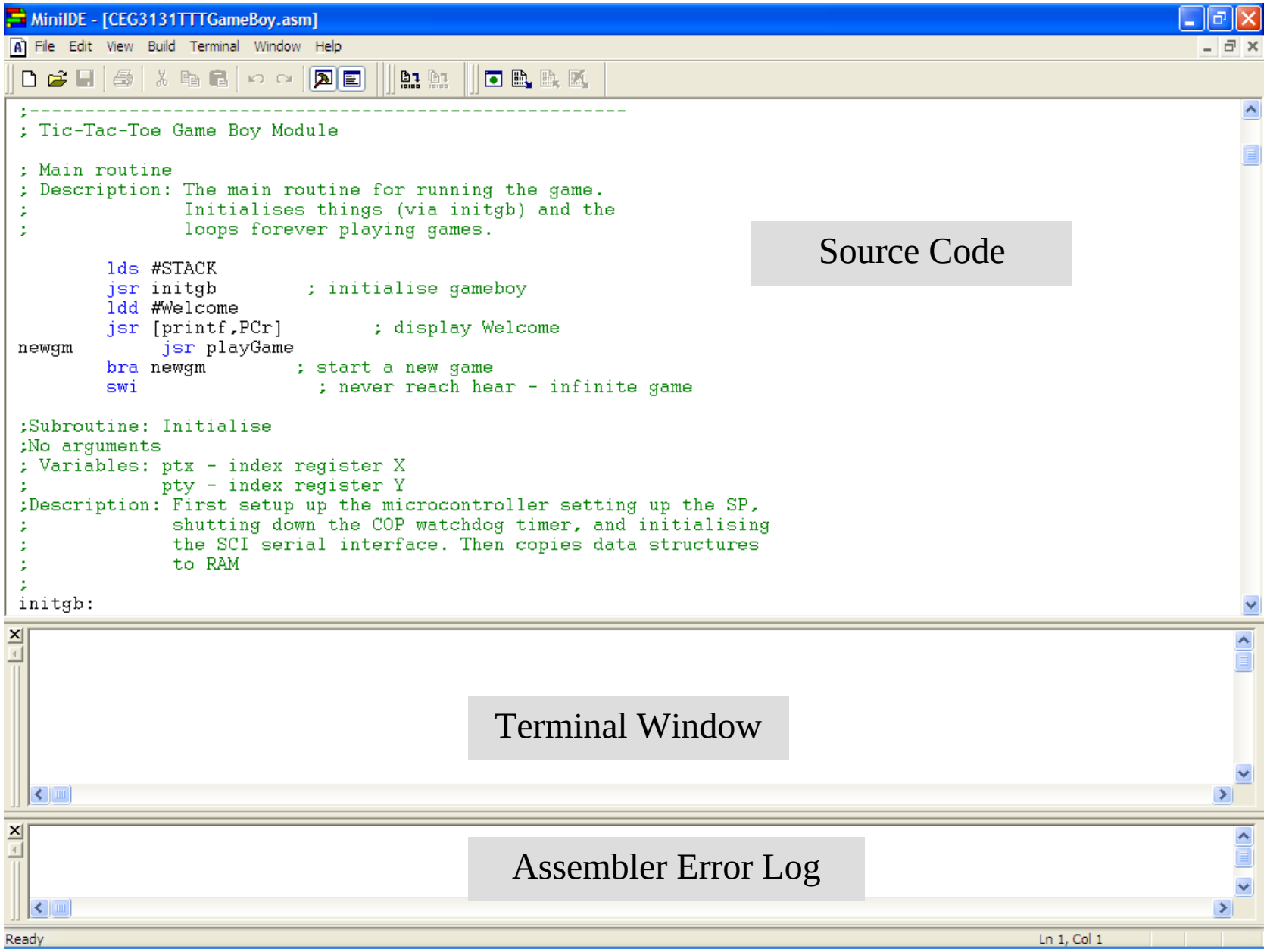
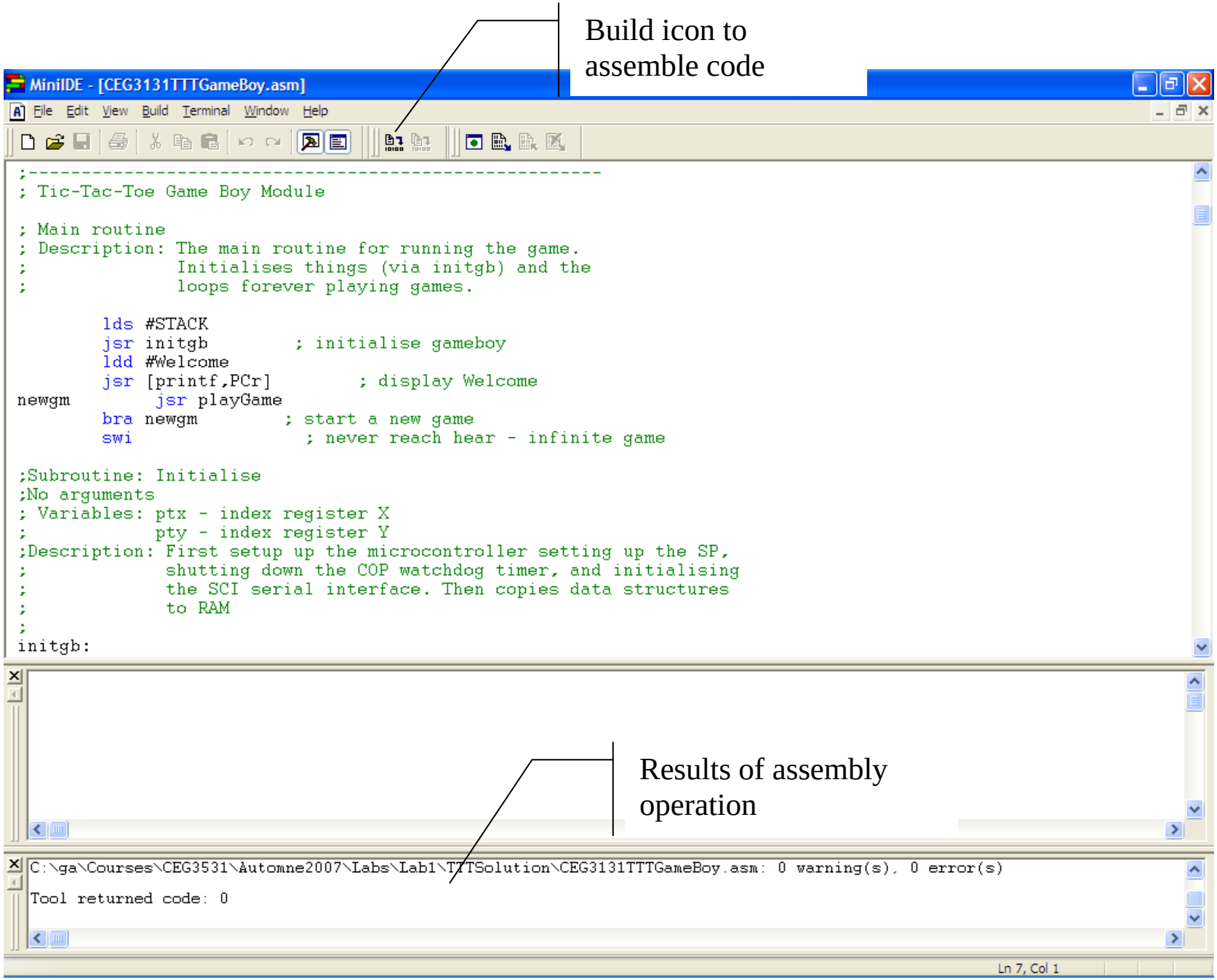
CEG 3136 – Computer Architecture II
Tutorial 1 - Introduction to Assembler Programming Fall 2019




MiniIDE - [CEG3131TTTGameBoy.asm]
File Edit View Build Terminal Window Help
;---
; Tic-Tac-Toe Game Boy Module
; Main routine
; Description: The main routine for running the game.
; Initialises things (via initgb) and the
; loops forever playing games.
lds #STACK
jsr initgb ; initialise gameboy
ldd #Welcome
jsr [printf,PCr] ; display Welcome
newgm jsr playGame
bra newgm ; start a new game
swi ; never reach hear - infinite game
;Subroutine: Initialise
;No arguments
; Variables: ptx - index register X
; pty - index register Y
;Description: First setup up the microcontroller setting up the SP,
; shutting down the COP watchdog timer, and initialising
; the SCI serial interface. Then copies data structures
; to RAM
;
initgb:
Source Code
Terminal Window
Assembler Error Log
Ready
Ln 1, Col 1


Build icon to assemble code
MiniIDE - [CEG3131TTTGameBoy.asm]
File Edit View Build Terminal Window Help
;---
; Tic-Tac-Toe Game Boy Module
; Main routine
; Description: The main routine for running the game.
; Initialises things (via initgb) and the
; loops forever playing games.
lds #STACK
jsr initgb ; initialise gameboy
ldd #Welcome
jsr [printf,PCr] ; display Welcome
newgm jsr playGame
bra newgm ; start a new game
swi ; never reach hear - infinite game
;Subroutine: Initialise
;No arguments
; Variables: ptx - index register X
; pty - index register Y
;Description: First setup up the microcontroller setting up the SP,
; shutting down the COP watchdog timer, and initialising
; the SCI serial interface. Then copies data structures
; to RAM
;
initgb:
Results of assembly operation
C:\ga\Courses\CEG3531\Automne2007\Labs\Lab1\TTTSolution\CEG3131TTTGameBoy.asm: 0 warning(s), 0 error(s)
Tool returned code: 0
Ln 7, Col 1

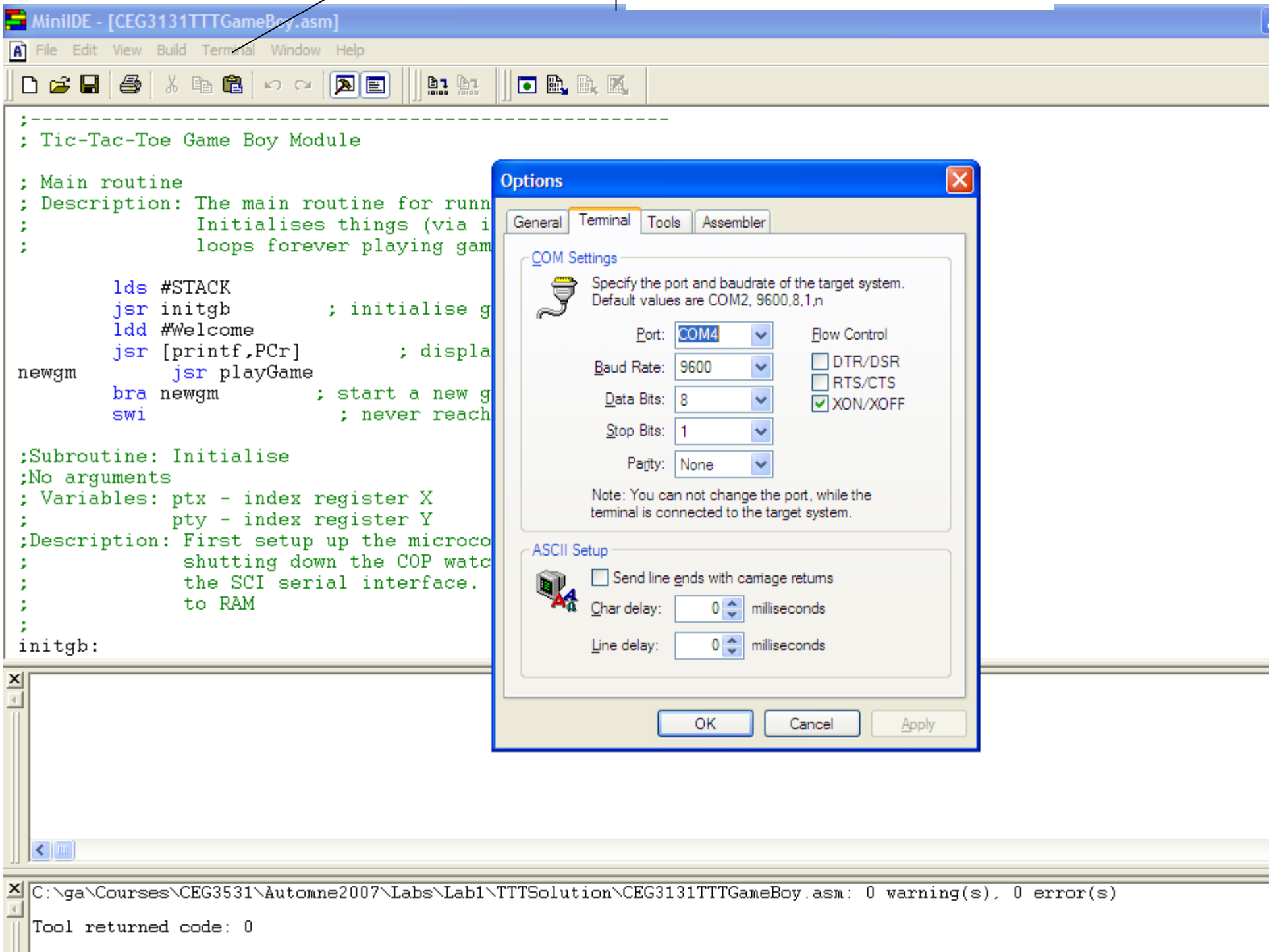
Contents of the S19 File

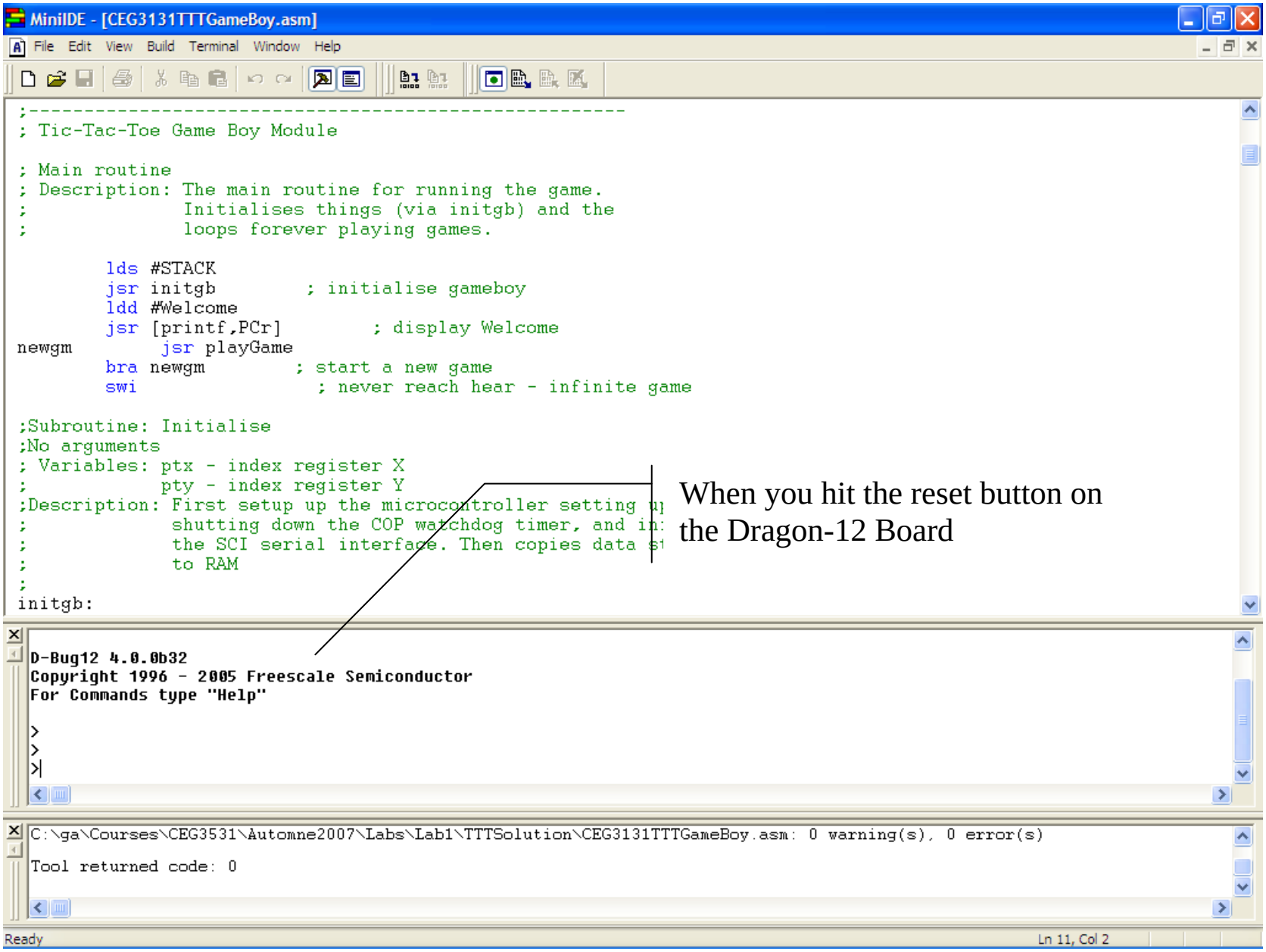
```
S0030000FC
S113040CF2000160413CC068115FBEA7B16045397
S113041020FB3F1410CE00000DE039800CE03A4080
S113042086056AE03486016AE0350FE03708FB0C84
     .
     .
S113056323DCE06BA0710B7101806180636CE0666
S113057CF07043218063D34EC8015FBE90A15FB5D
S113058E9023715FBE8FFCC071C15FBE8FA33C179
S113059302D04C1332D09CC06A415FBE8EA20D87C
     .
     .
S112071776F6E0A0D004472617721210A0D0084
S113250202020202020202020000053434F52452B
S113251A20506C6179657220583A20303020504E
S113252C61796572204F3A2030300A0D00506C8E
S113253617965722058206D6F76650A0D00203030
S113254202020312020320A0D2020207C202031
S113255207C20202020300A0D2D2D2D2B2D2D2DDB
S113256B2D2D2D0A0D2020207C2020207C2020A6
S113257202031A0D2D2D2D2B2D2D2D2B2D2D2DE4
S113258A0D2020207C2020207C20202020320ABC
S1052590D0038
S9030000FC
```

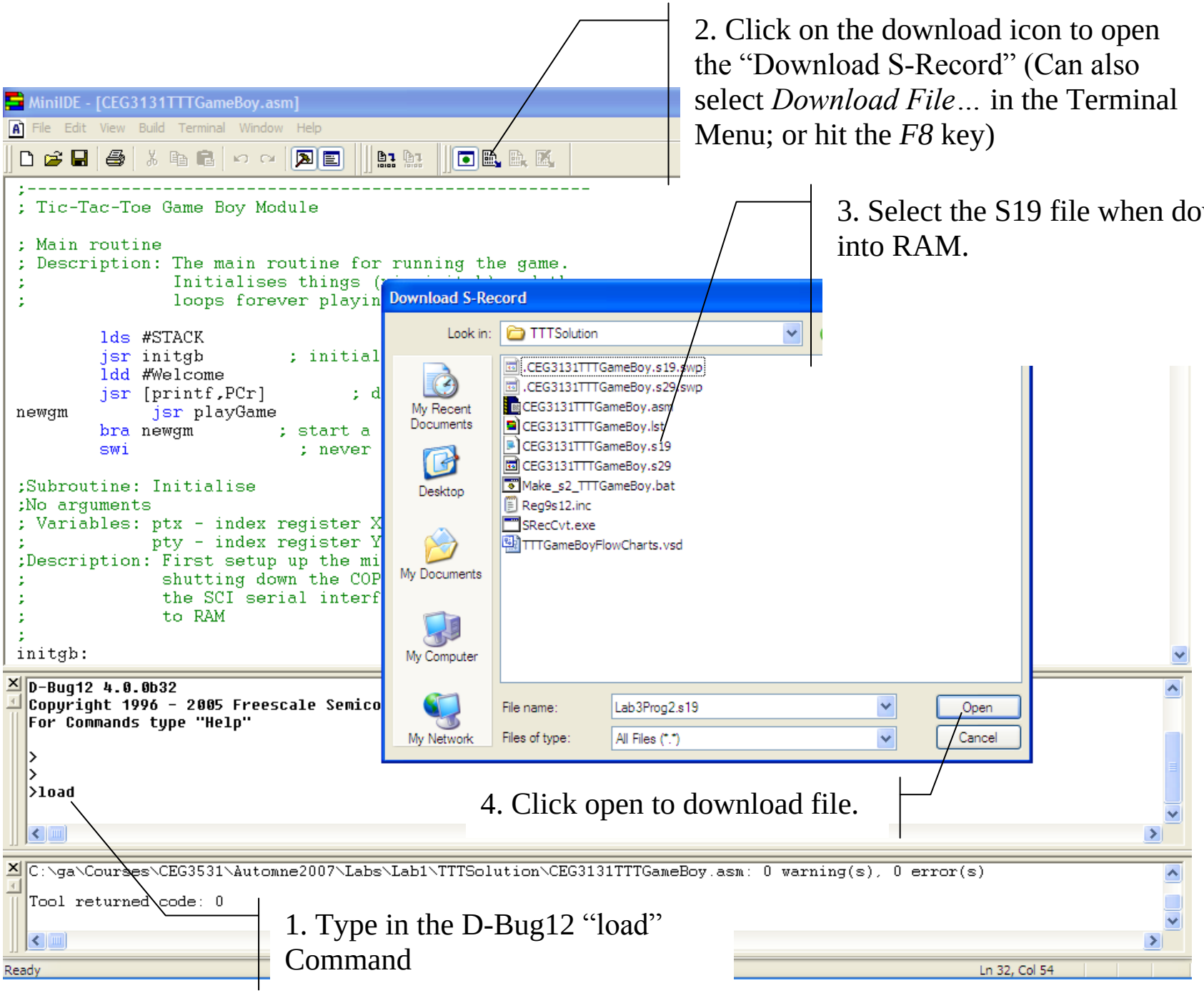
Contents of the S29 File

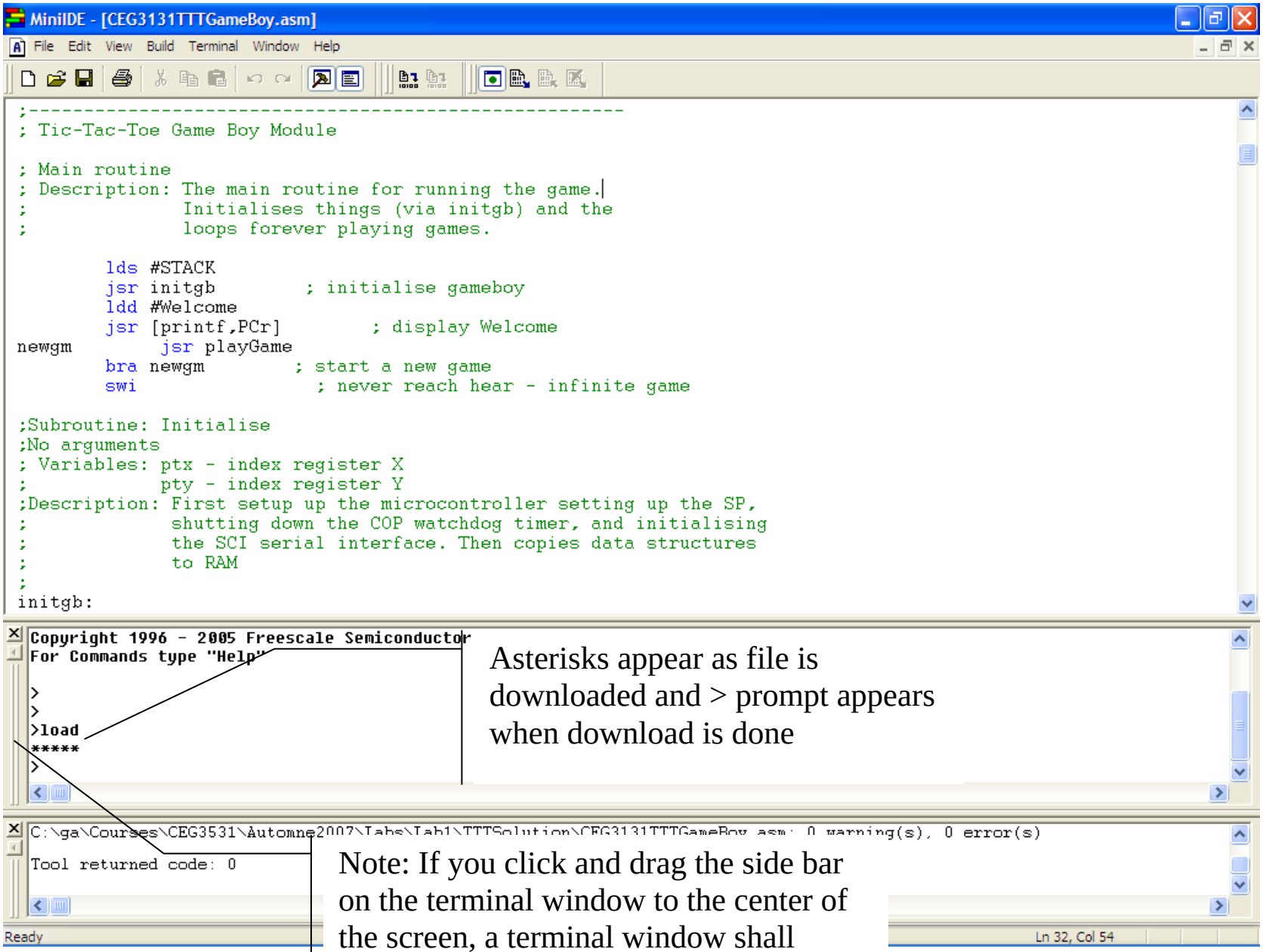
```
S2240F0400CF2000160413CC068115FBEA7B16045320FB3F1410CE00000DE039800CE03A401F
S2240F0420860056AE03486016AE0350FE03708FB0CE03980CC009C5CC8860C5ACBCE05F0CDF8
S2240F0440250B180A3070A60081FF26F679250979250A3DCE250034CC06E715FBEA2A1604AA
     .
     .
S2240F0560323DCE06BA0710B7101806180636CE06CF07043218063D34EC8015FBE90A15FB2C
S2240F0580E9023715FBE8FFCC071C15FBE8FA33C1302D04C1332D09CC06A415FBE8EA20D87E
S2240F05A0C030303DCE0678E63026040732200E180FC40F4444444405812027EA3D34CED0    .
     .
     .
S2240F07006572650A0D005820776F6E0A0D004F20776F6E0A0D004472617721210A0D00FFCF
S2240F2500202020202020202020000053434F52453A20506C6179657220583A2030302050A2
S2240F25206C61796572204F3A2030300A0D00506C617965722058206D6F76650A0D00203007
S2240F2540202020312020320A0D2020207C2020207C20202020300A0D2D2D2D2B2D2D2D75
S2240F25602B2D2D2D0A0D2020207C2020207C20202020310A0D2D2D2D2B2D2D2D2B2D2D2D13
S2240F25800A0D2020207C2020207C20202020320A0D00FFFFFFFFFFFFFFFFFFFFFFFFFF9D
S9030000FC
```

Select « Options » in Terminal menu to get the options dialog



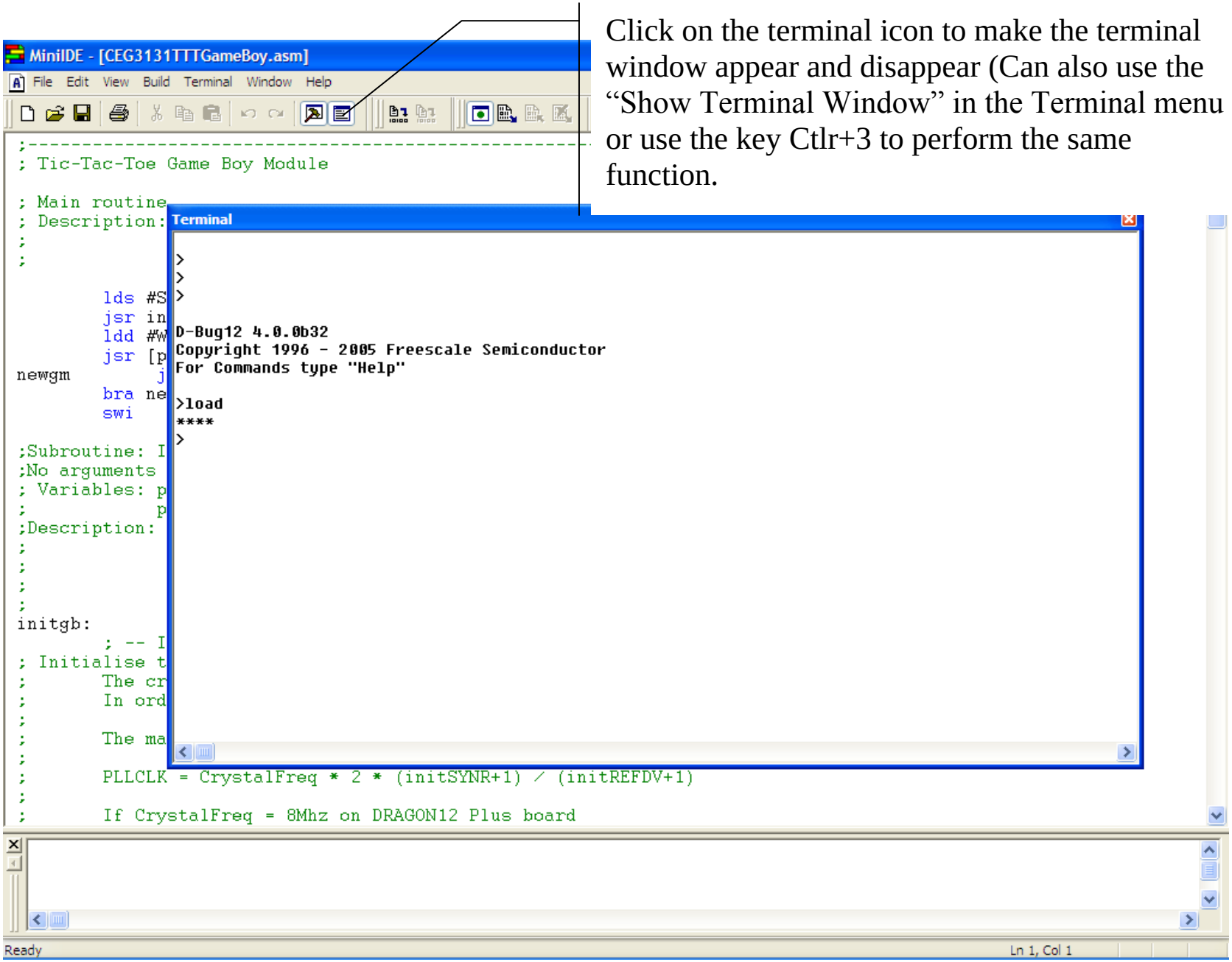

MiniIDE - [CEG3131TTTGameBoy.asm]
File Edit View Build Terminal Window Help
;---
; Tic-Tac-Toe Game Boy Module
; Main routine
; Description: The main routine for running the game.
; Initialises things (via initgb) and the
; loops forever playing games.
lds #STACK
jsr initgb ; initialise gameboy
ldd #Welcome
jsr [printf,PCr] ; display Welcome
newgm jsr playGame
bra newgm ; start a new game
swi ; never reach hear - infinite game
;Subroutine: Initialise
;No arguments
; Variables: ptx - index register X
; pty - index register Y
;Description: First setup up the microcontroller setting u
; shutting down the COP watchdog timer, and in
; the SCI serial interface. Then copies data s
; to RAM
;
initgb:
When you hit the reset button on the Dragon-12 Board
D-Bug12 4.0.0b32
Copyright 1996 - 2005 Freescale Semiconductor
For Commands type "Help"
>
>
>
C:\ga\Courses\CEG3531\Automne2007\Labs\Lab1\TTTSolution\CEG3131TTTGameBoy.asm: 0 warning(s), 0 error(s)
Tool returned code: 0
Ready
Ln 11, Col 2


2. Click on the download icon to open the "Download S-Record" (Can also select *Download File…* in the Terminal Menu; or hit the *F8* key)
3. Select the S19 file when downloading into RAM.
4. Click open to download file.
1. Type in the D-Bug12 "load" Command
MiniIDE - [CEG3131TTTGameBoy.asm]
File Edit View Build Terminal Window Help
; Tic-Tac-Toe Game Boy Module
; Main routine
; Description: The main routine for running the game.
lds #STACK
jsr initgb
ldd #Welcome
jsr [printf,PCr]
newgm jsr playGame
bra newgm
swi
;Subroutine: Initialise
;No arguments
initgb:
Download S-Record
Look in: TTTSolution
My Recent Documents
Desktop
My Documents
My Computer
My Network
.CEG3131TTTGameBoy.s19.swp
.CEG3131TTTGameBoy.s29.swp
CEG3131TTTGameBoy.asm
CEG3131TTTGameBoy.lst
CEG3131TTTGameBoy.s19
CEG3131TTTGameBoy.s29
Make_s2_TTTGameBoy.bat
Reg9s12.inc
SRecCvt.exe
TTTGameBoyFlowCharts.vsd
File name: Lab3Prog2.s19
Files of type: All Files (*.*)
Open
Cancel
D-Bug12 4.0.0b32
Copyright 1996 - 2005 Freescale Semico
For Commands type "Help"
>load
C:\ga\Courses\CEG3531\Automne2007\Labs\Lab1\TTTSolution\CEG3131TTTGameBoy.asm: 0 warning(s), 0 error(s)
Tool returned code: 0
Ready
Ln 32, Col 54



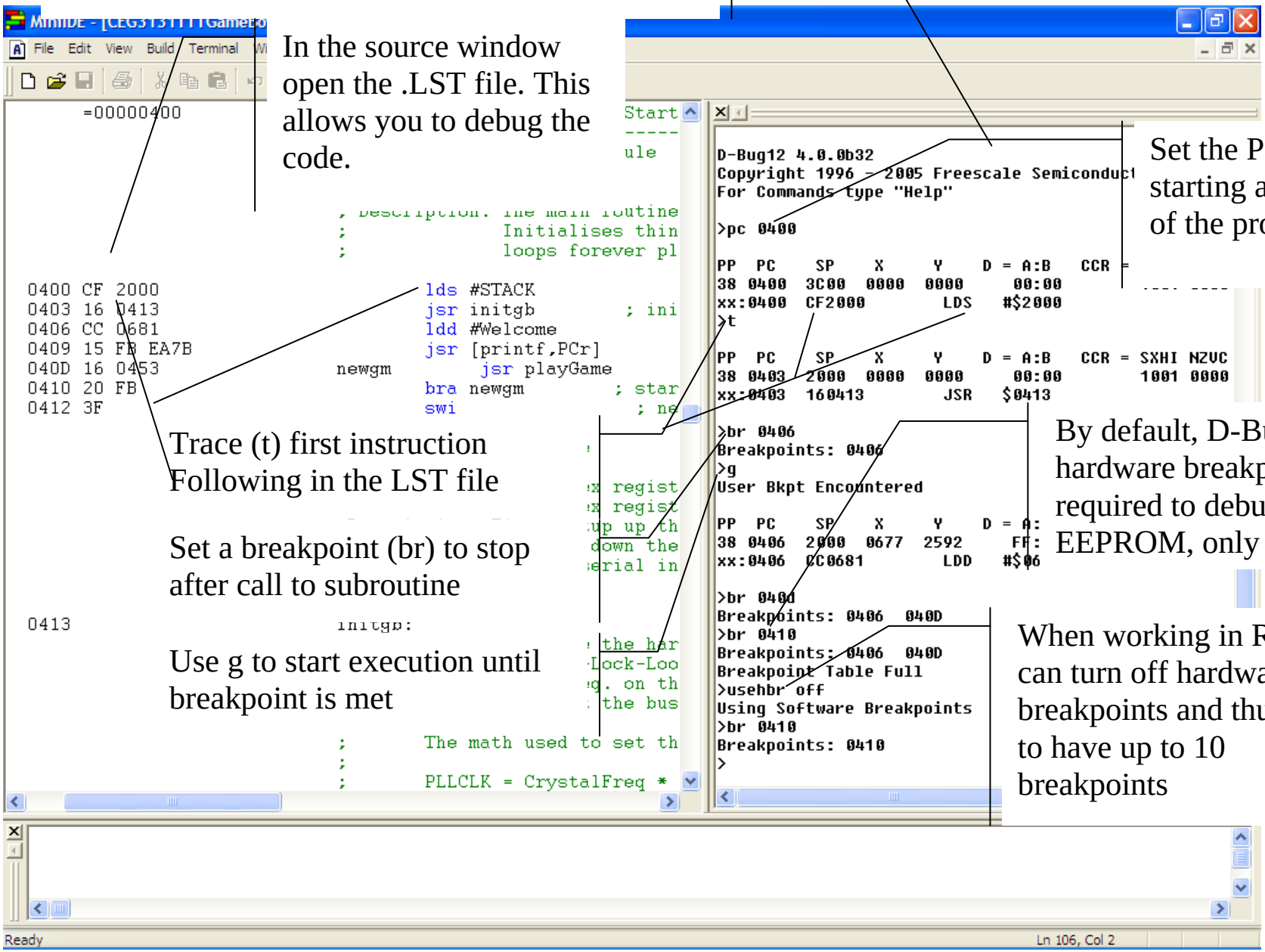
Asterisks appear as file is downloaded and > prompt appears when download is done

Note: If you click and drag the side bar on the terminal window to the center of the screen, a terminal window shall appear as on the next page



Click on the terminal icon to make the terminal window appear and disappear (Can also use the “Show Terminal Window” in the Terminal menu or use the key Ctlr+3 to perform the same function.

Another placement of the terminal window. Drag the terminal window to the right edge of the IDE.



In the source window open the .LST file. This allows you to debug the code.

Set the PC to the starting address of the program

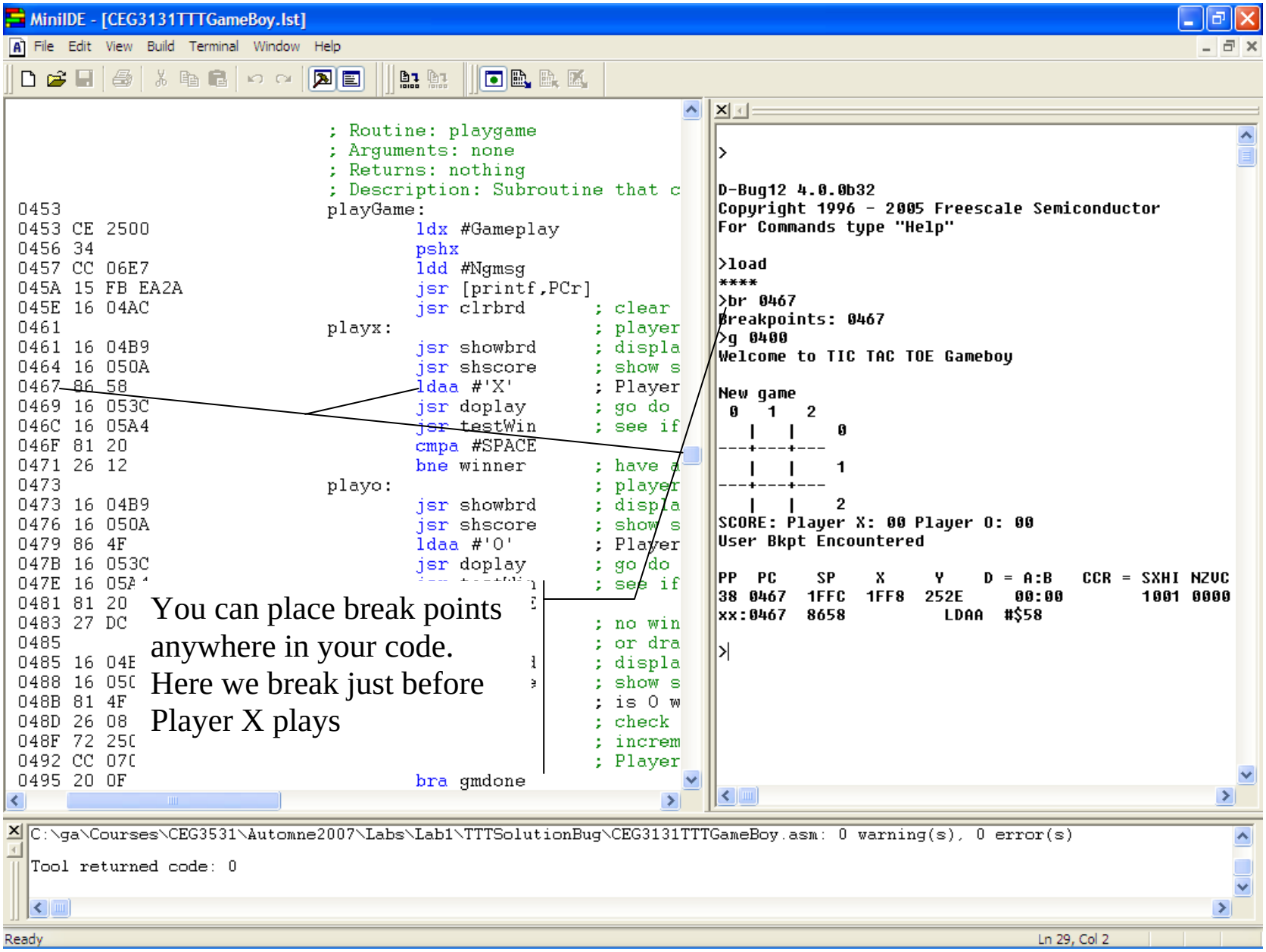
Trace (t) first instruction Following in the LST file

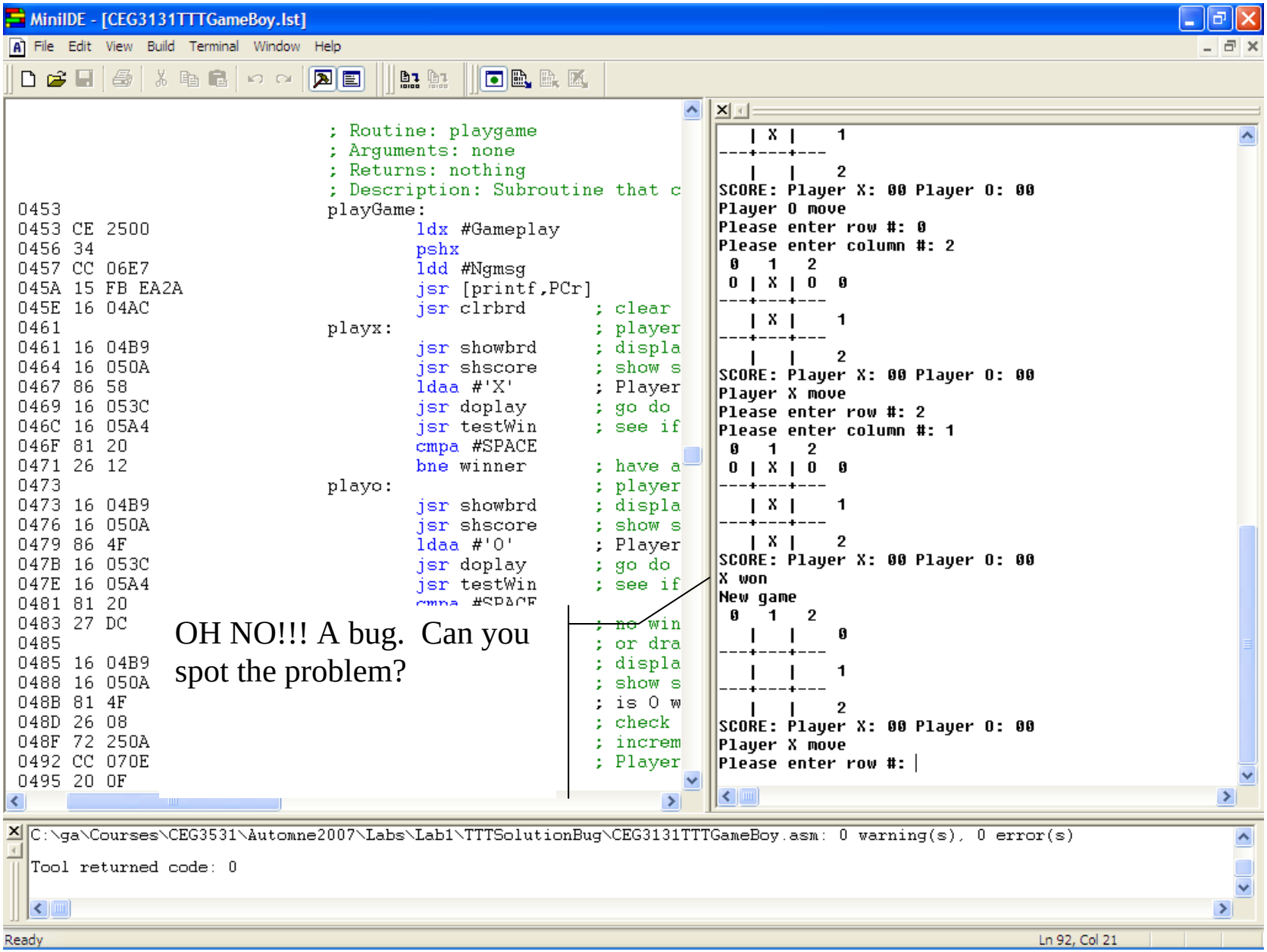
By default, D-Bug12 uses hardware breakpoints required to debug code in EEPROM, only 2 allowed.

Set a breakpoint (br) to stop after call to subroutine

Use g to start execution until breakpoint is met

When working in RAM, can turn off hardware breakpoints and thus use up to have up to 10 breakpoints


MiniIDE - [CEG3131TTTGameBoy.lst]
File Edit View Build Terminal Window Help
; Routine: playgame
; Arguments: none
; Returns: nothing
; Description: Subroutine that c
0453 playGame:
0453 CE 2500 ldx #Gameplay
0456 34 pshx
0457 CC 06E7 ldd #Ngmsg
045A 15 FB EA2A jsr [printf,PCr]
045E 16 04AC jsr clrbrd ; clear
0461 playx: ; player
0461 16 04B9 jsr showbrd ; displa
0464 16 050A jsr shscore ; show s
0467 86 58 ldaa #'X' ; Player
0469 16 053C jsr doplay ; go do
046C 16 05A4 jsr testWin ; see if
046F 81 20 cmpa #SPACE
0471 26 12 bne winner ; have a
0473 playo: ; player
0473 16 04B9 jsr showbrd ; displa
0476 16 050A jsr shscore ; show s
0479 86 4F ldaa #'O' ; Player
047B 16 053C jsr doplay ; go do
047E 16 05A ; see if
0481 81 20
0483 27 DC ; no win
0485 ; or dra
0485 16 04E ; displa
0488 16 050 ; show s
048B 81 4F ; is O w
048D 26 08 ; check
048F 72 250 ; increm
0492 CC 070 ; Player
0495 20 0F bra gmdone
You can place break points anywhere in your code. Here we break just before Player X plays
>
D-Bug12 4.0.0b32
Copyright 1996 - 2005 Freescale Semiconductor
For Commands type "Help"
>load
****
>br 0467
Breakpoints: 0467
>g 0400
Welcome to TIC TAC TOE Gameboy
New game
0 1 2
| | 0
---+---+---
| | 1
---+---+---
| | 2
SCORE: Player X: 00 Player O: 00
User Bkpt Encountered
PP PC SP X Y D = A:B CCR = SXHI NZVC
38 0467 1FFC 1FF8 252E 00:00 1001 0000
xx:0467 8658 LDAA #$58
>
C:\ga\Courses\CEG3531\Automne2007\Labs\Lab1\TTTSolutionBug\CEG3131TTTGameBoy.asm: 0 warning(s), 0 error(s)
Tool returned code: 0
Ready
Ln 29, Col 2


MiniIDE - [CEG3131TTTGameBoy.lst]
File Edit View Build Terminal Window Help
; Routine: playgame
; Arguments: none
; Returns: nothing
; Description: Subroutine that c
0453 playGame:
0453 CE 2500 ldx #Gameplay
0456 34 pshx
0457 CC 06E7 ldd #Ngmsg
045A 15 FB EA2A jsr [printf,PCr]
045E 16 04AC jsr clrbrd ; clear
0461 playx: ; player
0461 16 04B9 jsr showbrd ; displa
0464 16 050A jsr shscore ; show s
0467 86 58 ldaa #'X' ; Player
0469 16 053C jsr doplay ; go do
046C 16 05A4 jsr testWin ; see if
046F 81 20 cmpa #SPACE
0471 26 12 bne winner ; have a
0473 playo: ; player
0473 16 04B9 jsr showbrd ; displa
0476 16 050A jsr shscore ; show s
0479 86 4F ldaa #'O' ; Player
047B 16 053C jsr doplay ; go do
047E 16 05A4 jsr testWin ; see if
0481 81 20
0483 27 DC ; no win
0485 ; or dra
0485 16 04B9 ; displa
0488 16 050A ; show s
048B 81 4F ; is O w
048D 26 08 ; check
048F 72 250A ; increm
0492 CC 070E ; Player
0495 20 0F
OH NO!!! A bug. Can you spot the problem?
| X | 1
---+---+---
| | 2
SCORE: Player X: 00 Player O: 00
Player O move
Please enter row #: 0
Please enter column #: 2
0 1 2
0 | X | O 0
---+---+---
| X | 1
---+---+---
| | 2
SCORE: Player X: 00 Player O: 00
Player X move
Please enter row #: 2
Please enter column #: 1
0 1 2
0 | X | O 0
---+---+---
| X | 1
---+---+---
| X | 2
SCORE: Player X: 00 Player O: 00
X won
New game
0 1 2
| | 0
---+---+---
| | 1
---+---+---
| | 2
SCORE: Player X: 00 Player O: 00
Player X move
Please enter row #:
C:\ga\Courses\CEG3531\Automne2007\Labs\Lab1\TTTSolutionBug\CEG3131TTTGameBoy.asm: 0 warning(s), 0 error(s)
Tool returned code: 0
Ready
Ln 92, Col 21

This subroutine contains a bug

```
; Subroutine - shscore
; Arguments: none
; Returns: nothing
; Global Variables: Xscr - X score number
;                   Xscr_a - X score ASCII
;                   Oscr - O score number
;                   Oscr_a - O score ASCII
; Description: Outputs current score, but first
;    translates the hexadecimal value to a 2 byte
;    ASCII value. Value is passed in Acc D to scr2asc
;    for translation. Register X contains address
;    where converted bytes are to be stored.
;    After scores are converted, score string is displayed.
shscore      pshd
       pshx   ; preserve registers
       ldab Xscr
       clra
       ldx #Xscr_a
       bsr scr2asc
       ldab Oscr
       clra
       ldx #Oscr_a
       bsr scr2asc
       ldd #Score
       jsr [printf,PCr] ; display current Score
       pulx   ; restore registers
       puld
       rts
; Subroutine; scr2asc (num,addr)
; Parameters: num - in accumulator D (need in D for divide)
;             addr - in Y register
; Local variables: rem - in accumulator D
;                  qu - in X register
; Converts number to ascii decimal equivalent
scr2asc
        pshx
        pshd        ; preserve
        ldx #10
        idiv                ; remainder in D, i.e. B
        addb #ASCCONVNUM ; converts digit to ASCII
        stab 1,y          ; save first digit
        tfr x,d                ; move quotient back to d
        addb #ASCCONVNUM ; convert digit to ASCII
        stab 0,y          ; save second digit
        puld        ; restore
        pulx
        rts
```

Basic Programming

Questions:

1. What is the meaning of the sign bit = 1 when unsigned binary coded numbers are added?

2. What is the meaning of the carry bit = 1 when two unsigned binary coded numbers are subtracted? When two's complement binary coded numbers are subtracted?

3. What addressing mode is best to use when you want to access several sequential elements in a data array – immediate, direct, indexed?

4. Pointer addressing with auto-increment and auto-decrement is referred to as what type of addressing for the MC68HC12?

Exercise 1

Give the values of the starting address, offset (indicate the size of the offset) and calculate the effective address of each of the following examples. Illustrate on the programmer's model the effect of the instructions to the CPU registers and memory.

a. X = $3000 A=$E5
LDY A,X EA =

b. Y = $6F40
STD 4,-Y EA =

c. SP = $0BF3
ADDA 3,SP EA =

Exercise 2:

An array of bytes contains a set of non-zero unsigned values. The last value in the array contains the value 0 to indicate the end of the array. Write a piece of code that contains a loop to read the contents of each byte into accumulator A. The array starts at address ARRAY. Write a first version that uses indexed addressing with accumulator B as the offset. Write a second version that auto-increments the index register. (Use the programming model to follow the steps used in running the program for this first version and if time permits the second version).

Exercise 3

Consider two three byte numbers stored at addresses NUM1 and NUM2 as defined by the following assembler pseudo-operations:

```
NUM1    EQU    $0850
NUM2    EQU    $0853
DIFF    EQU    $0856
```

It is possible to load into the accumulators the bytes (or store into memory) using the labels as follows:

```
LDAA    NUM1      ; Loads most significant byte into accumulator A
STAB    DIFF+2    ; Store the contents of accumulator B into the least
                  ; Significant byte of the SUM number
```

The assembler will translate the labels to the appropriate addresses as shown below:

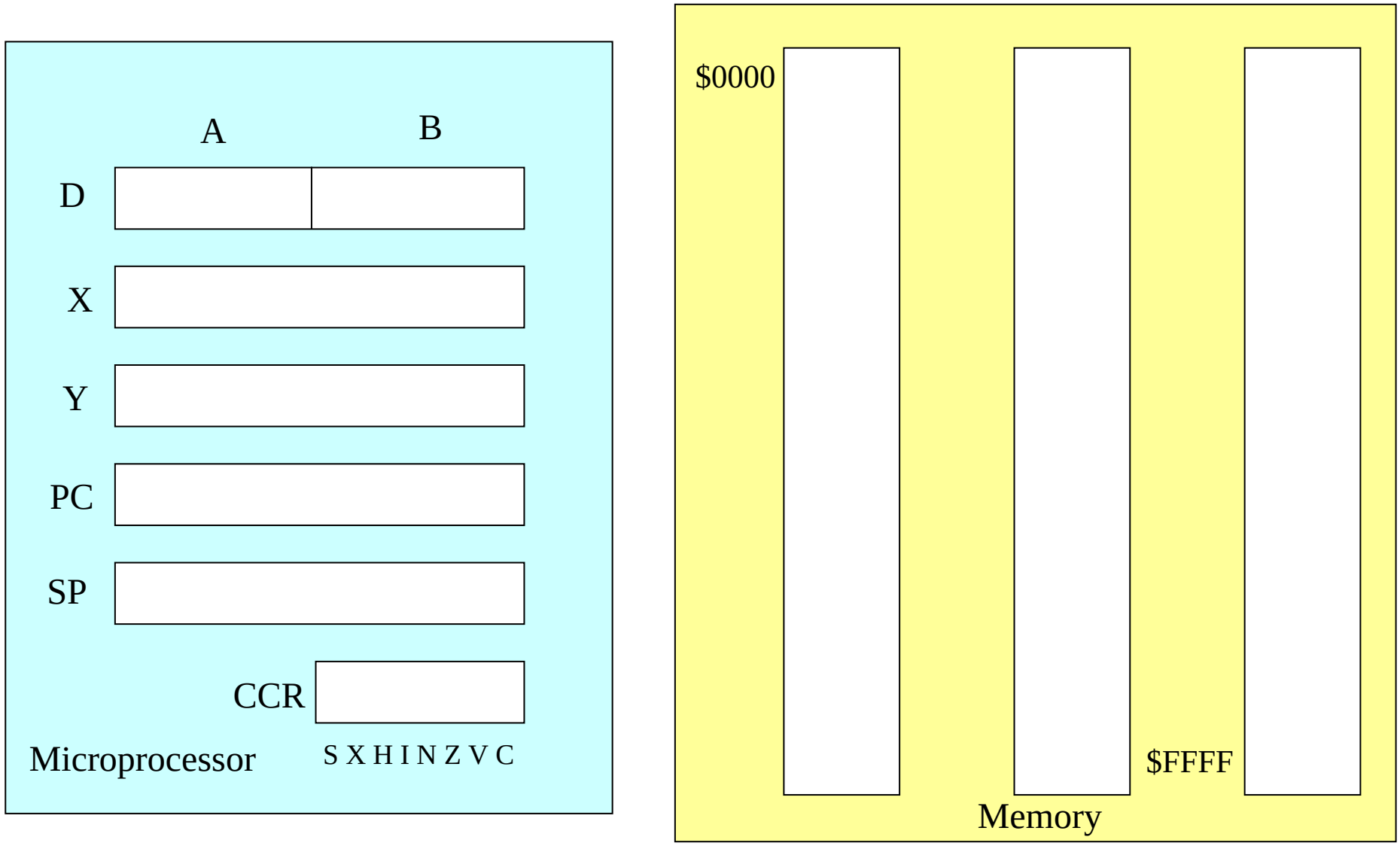
```
LDAA    $0850     ; NUM1
STAB    $0858     ; DIFF+2
```

The objective of this exercise is to write assembler source code to compute the difference between the two three-byte numbers (NUM1-NUM2) and store the result at address DIFF.

You will need the following 68HC12 instructions:

LDAA: Load accumulator A
SUBA: Subtract memory from A: A <- A- (M)
SBCA: Subtract memory and carry from A: A <- A – (M) - C
STAA: Store accumulator A

Use the programming model to follow the steps used in running the program.


A
B
D
X
Y
PC
SP
CCR
S X H I N Z V C
Microprocessor
$0000
$FFFF
Memory