A Web-Based Graphical Education Support Tool for Teaching the Course of Assembly Language Programming

URL: http://cis.k.hosei.ac.jp/~yamin/asm/

Yamin Li, Department of Computer Science, Hosei University, yamin@hosei.ac.jp

Assembly Language Programming

- The foundation of many machines, from general-purpose computer systems to embedded systems, lies in assembly language programming and computer architecture
  - If someone doesn't understand assembly language, he (she) can neither develop a compiler nor design a CPU
- The assembly language can be used to demonstrate what is actually happening inside the computer
  - It is an important path to understanding how the computer works at the machine level

Assembly Language Programming

- Some special (hardware) operations may not be controlled in C or Java
  - Example: How to select one of the following round-modes of FPU in C?
    - Round to nearest (to even if g r s = 1 0 0)
    - Round toward minus infinity ($-\infty$)
    - Round toward plus infinity ($+\infty$)
    - Round toward zero (also called truncate)
- Some assembly codes must be inserted in C

```
#define Near asm volatile("fldcw _RoundNear")
int _RoundNear = 0x103f; // Round mode = 00
Near; // Round to nearest
```

Top Three Assembly Programming Languages

- High-level languages are machine independent
- Assembly languages are machine dependent
- Top three assembly programming languages
  - Intel x86
  - ARM
  - MIPS
- We selected MIPS instruction set
  - MIPS is a typical RISC instruction set
  - To design MIPS CPUs in the following course of Computer Organization and Design

Functions of Simulators for Assembly Programming

- The basic assembly programs' assembling and simulation:
  - Grammar checking
  - Translating to binary code
  - Single-step execution
  - Break-point setting
  - Showing the contents of the register file
  - Showing the contents of the data memory
- The basic input and output system function calls, like scanf() and printf() in C, or readLine() and System.out.print() in Java

Two Existing MIPS Simulators

- SPIM — Windows (PCSpim) and Linux (spim and xspim)
  - Register display that shows the values of registers
  - Control buttons that let us interact with simulator
  - Text segment that display instructions
  - Data and stack segment
  - Spim messages used to write messages
- MARS — Java-based simulator
  - Provided a self-contained program editor
  - Showed the displacements with labels
  - Changed the simulator layout

AsmSim — A MIPS Simulator We Developed

- Web-based — does not need installation
- Graphical — supports VGA
- The AsmSim provides three windows:
  - A main window that provides control buttons and displays the instructions, the values of registers, and the data of the memory
  - A graphics console window that is used to input command, to display messages, and to show the images of Video RAM (VRAM)
  - An online editor window for editing the assembly program

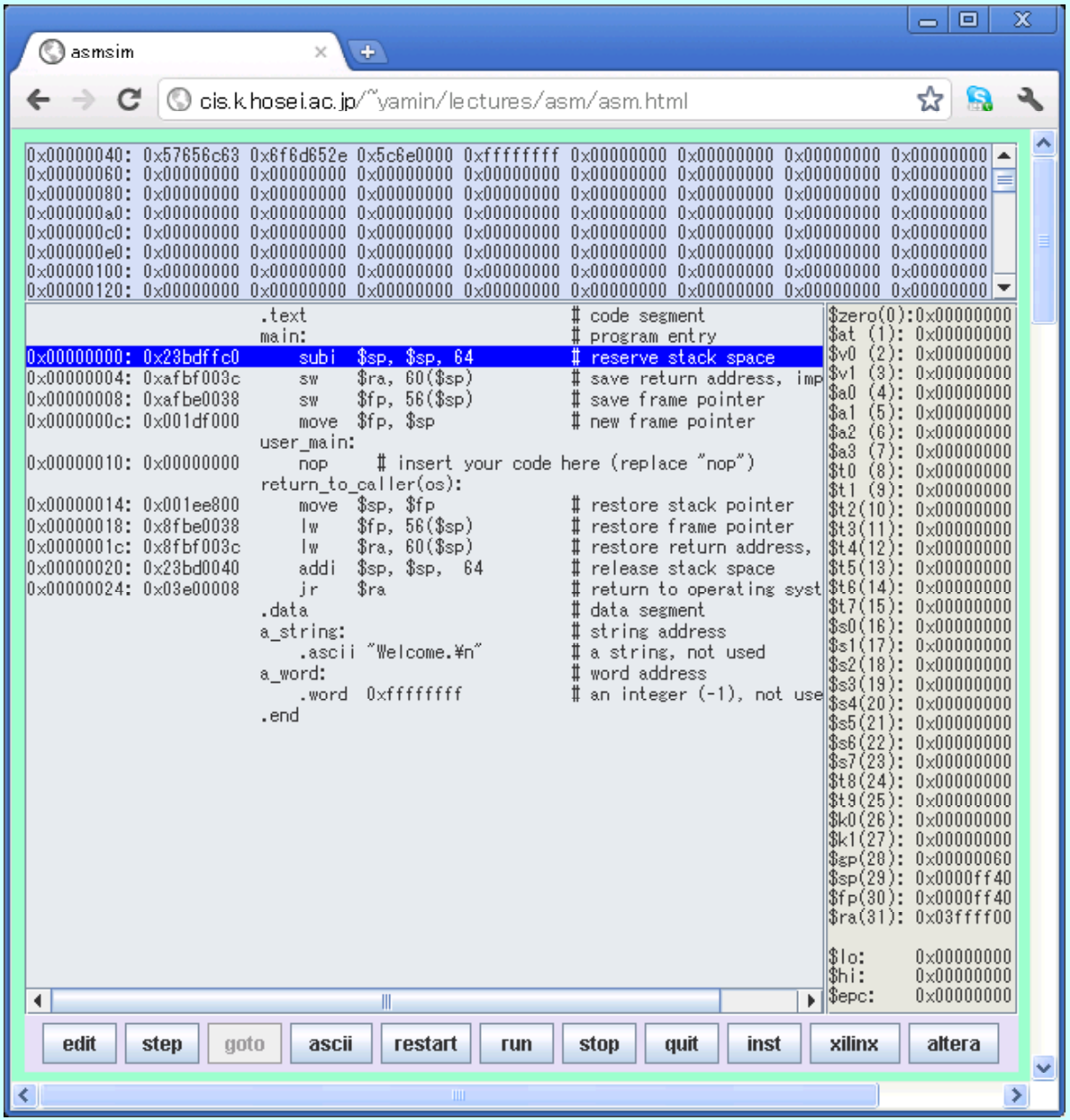
An Assembly Program Example

```
/*
 * hello.s
 * prints "Hello, World!" to console window
 * input: $4: the string address
 */
.text                           # code segment
main:                           # program entry
    subi   $sp, $sp, 24         # reserve stack space
    sw     $ra, 20($sp)         # save return address
    sw     $fp, 16($sp)         # save frame pointer
    move   $fp, $sp             # new frame pointer
user_main:
    la     $4,  msg             # hello message address
    jal    printf               # print out message
return_to_caller:               # exit()
    move   $sp, $fp             # restore stack pointer
    lw     $fp, 16($sp)         # restore frame pointer
    lw     $ra, 20($sp)         # restore return address
    addu   $sp, $sp, 24         # release stack space
    jr     $ra                  # return to operating system
.data                           # data segment
msg:                            # the string address
    .ascii "Hello, World!\n"    # the string
.end
```

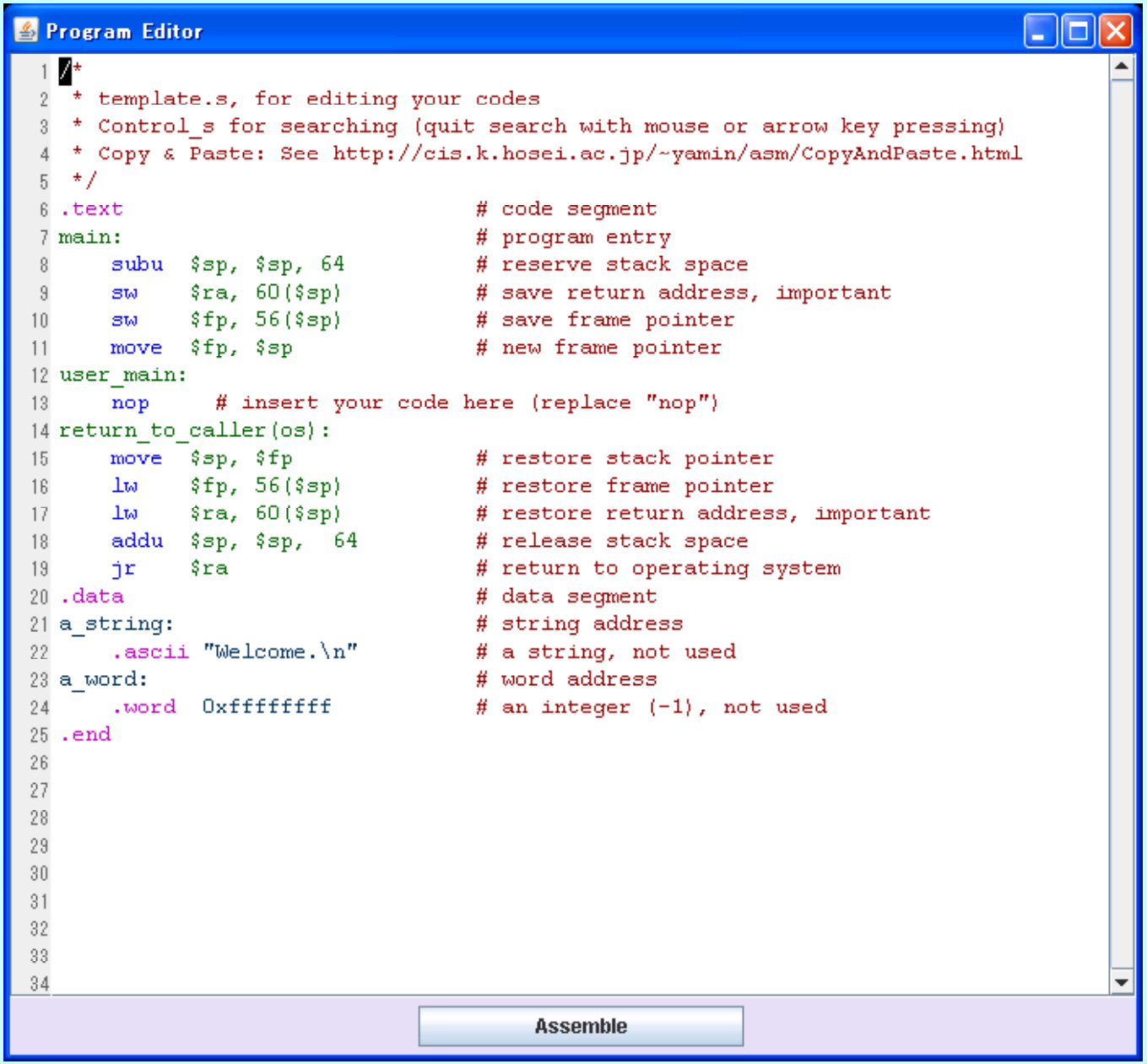
The Functions of AsmSim

- The basic assembling and simulation
- The basic input and output system function calls
- Interrupt mechanism for handling the keyboard interrupt
- Manipulating a standard VGA (640 $\times$ 480 pixels)
- Graphics object draw and fill
- Generating Xilinx COE and Altera MIF for FPGAs
- Some other useful function calls, like get a random number, get the timer, sleep, and get calendar
- Supporting the data structure of the linked-list for implementing some graph algorithms

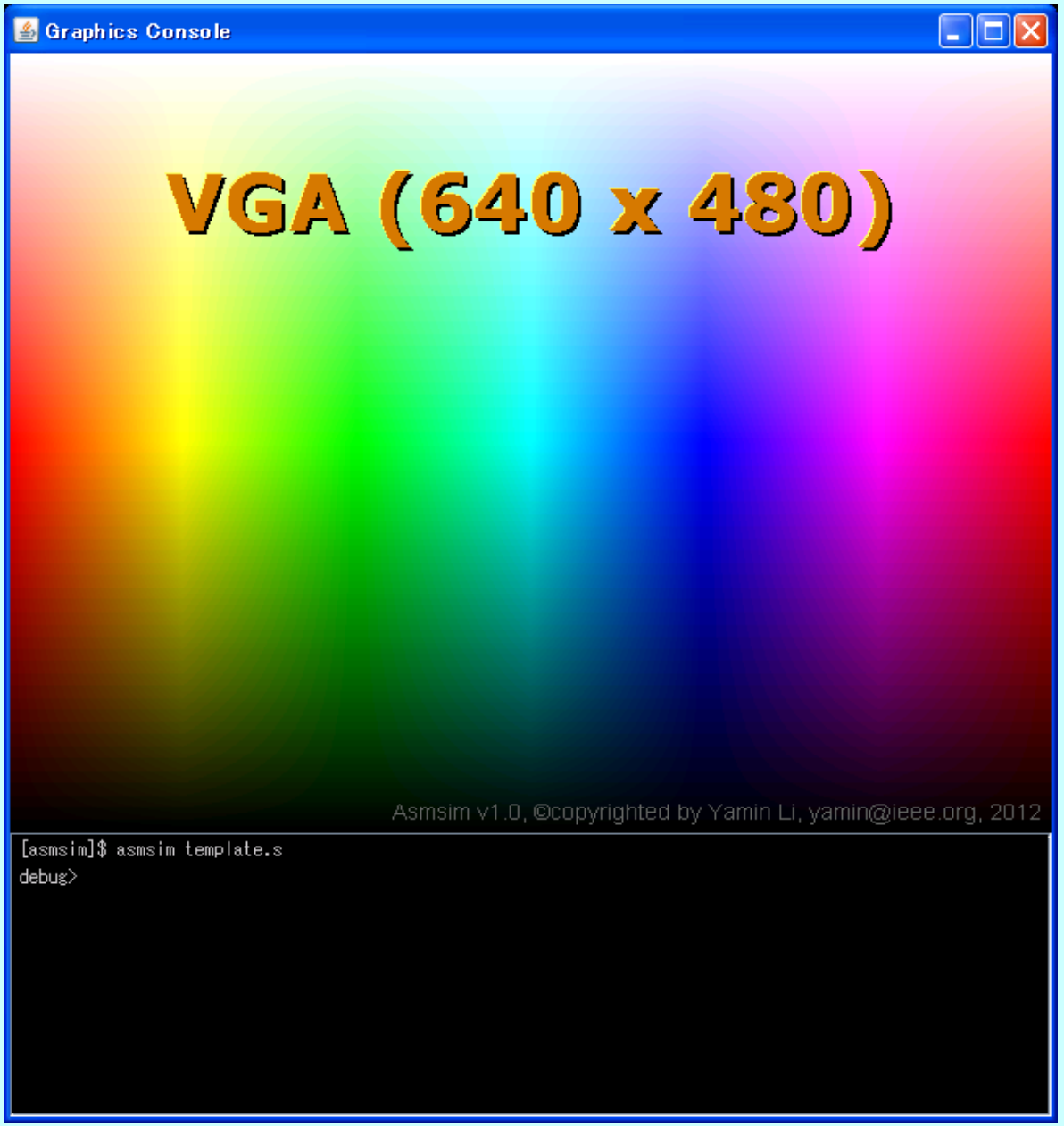
The Main Window of AsmSim



The Highlighted Program Edit Window of AsmSim



The Graphics Console Window of AsmSim



11 Buttons in the Main Window

1. **(edit)**: opens the program edit window
2. **(step)**: executes one instruction that is highlighted
3. **(goto)**: executes instructions to the break-point
4. **(ascii)**: displays the ascii of selected data
5. **(restart)**: re-loads the user program
6. **(run)**: executes user program
7. **(stop)**: stops the execution
8. **(quit)**: quits and enters *command line mode*
9. **(inst)**: displays the encodes of the MIPS instructions
10. **(xilinx)**: generates Xilinx COE file
11. **(altera)**: generates Altera MIF file

Instructions AsmSim Supported

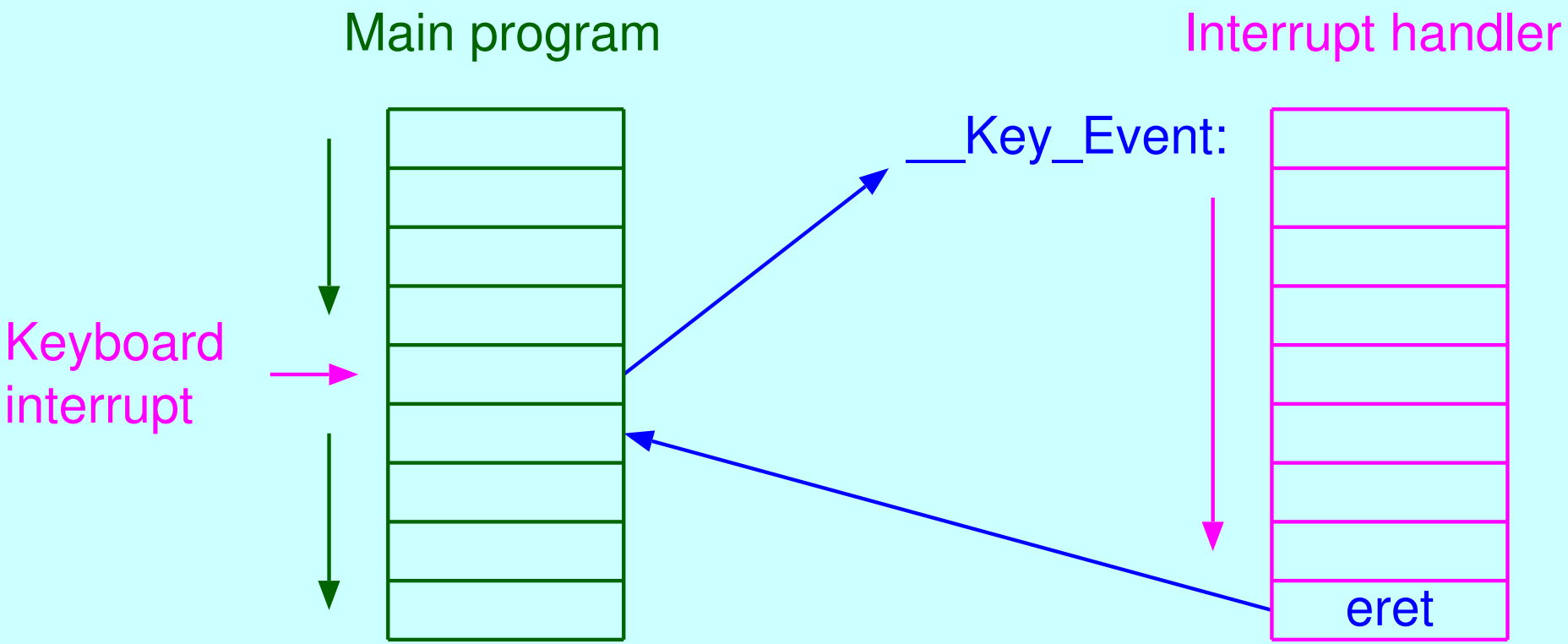
System Calls AsmSim Supported

1. **getchar()** gets a character from STDIN
2. **putchar()** puts a character to STDOUT
3. **scanf()** reads an input from STDIN
4. **printf()** writes output(s) to STDOUT
5. **sprintf()** writes output(s) to a buffer
6. **malloc()** allocates a block of memory
7. **gettimer()** gets the system timer in ms
8. **getrandom()** gets a random integer number
9. **getarrow()** gets an arrow key's information
10. **getcal()** gets the calendar in integers
11. **getcals()** gets the calendar in string
12. **sleep()** suspends execution for an interval of time
13. **paint()** shows the image on the VGA

System Calls AsmSim Supported

14. **drawline()** draws a color line
15. **drawoval()** draws a color oval
16. **drawrect()** draws a color rectangle
17. **drawrect3d()** draws a color 3D rectangle
18. **drawstring()** draws specified text
19. **filloval()** fills a color oval
20. **fillrect()** fills a color rectangle
21. **fillrect3d()** fills a color 3D rectangle
22. **refresh_vga_manu()** refreshes VGA by paint()
23. **refresh_vga_auto()** refreshes VGA automatically
24. **key_event_ena()** enables keyboard interrupt
25. **key_event_dis()** disables keyboard interrupt

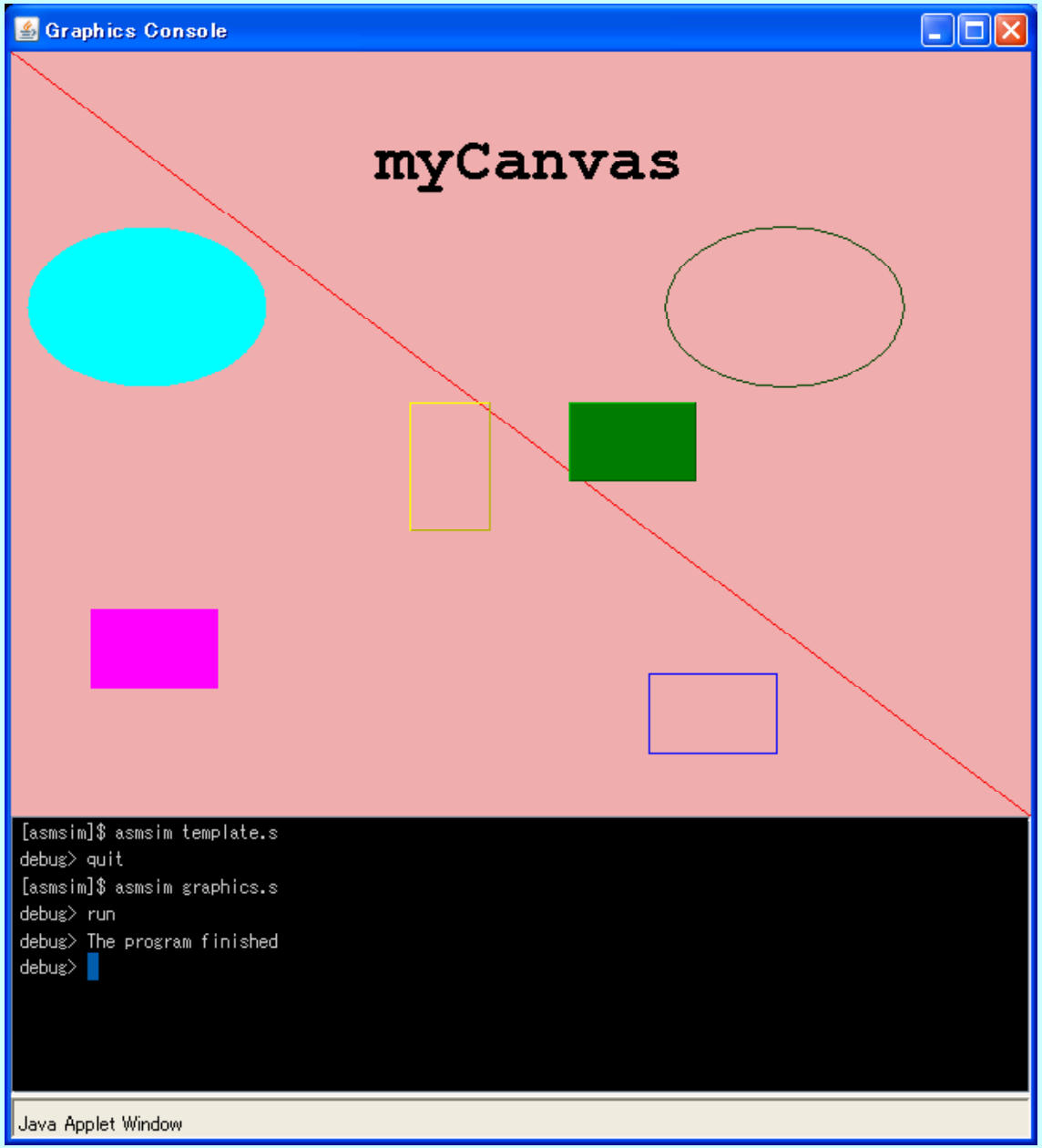
Interrupt Supported in AsmSim



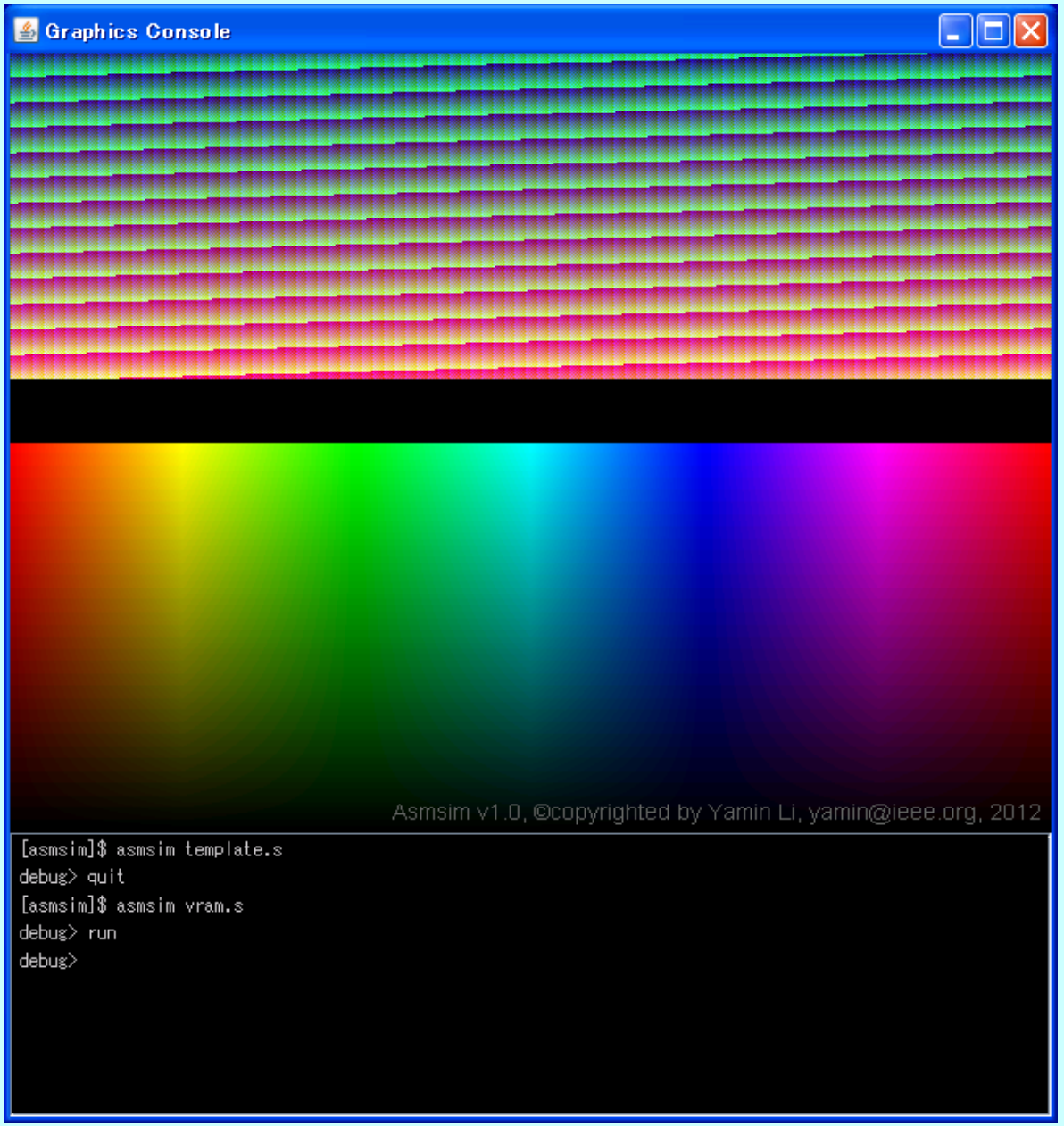
Before transfering to __Key_Event, EPC ← return address

eret (return from exception): PC ← EPC

The Graphics Output Example (graphics.s)

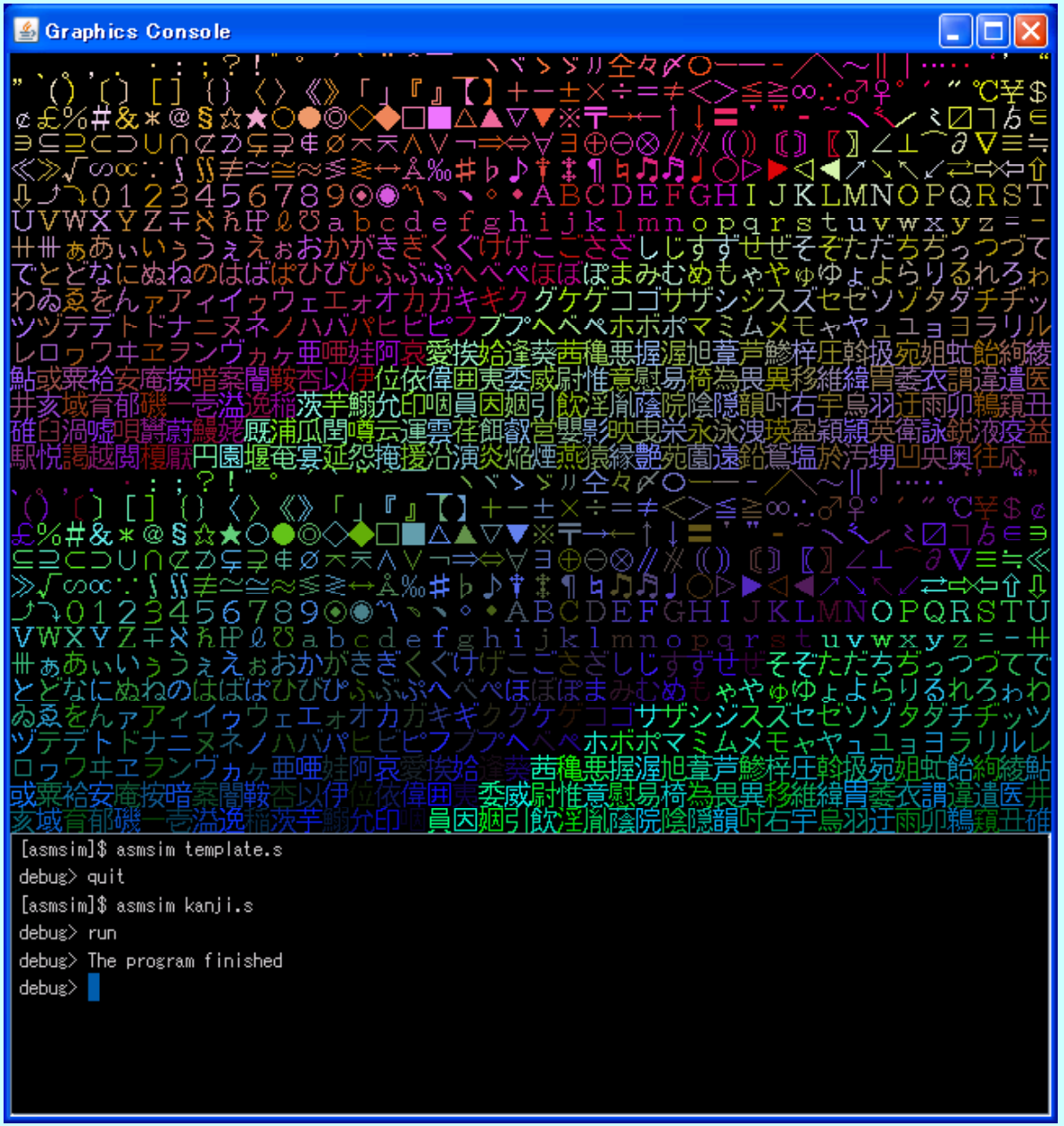


The Graphics Output Example (vram.s)

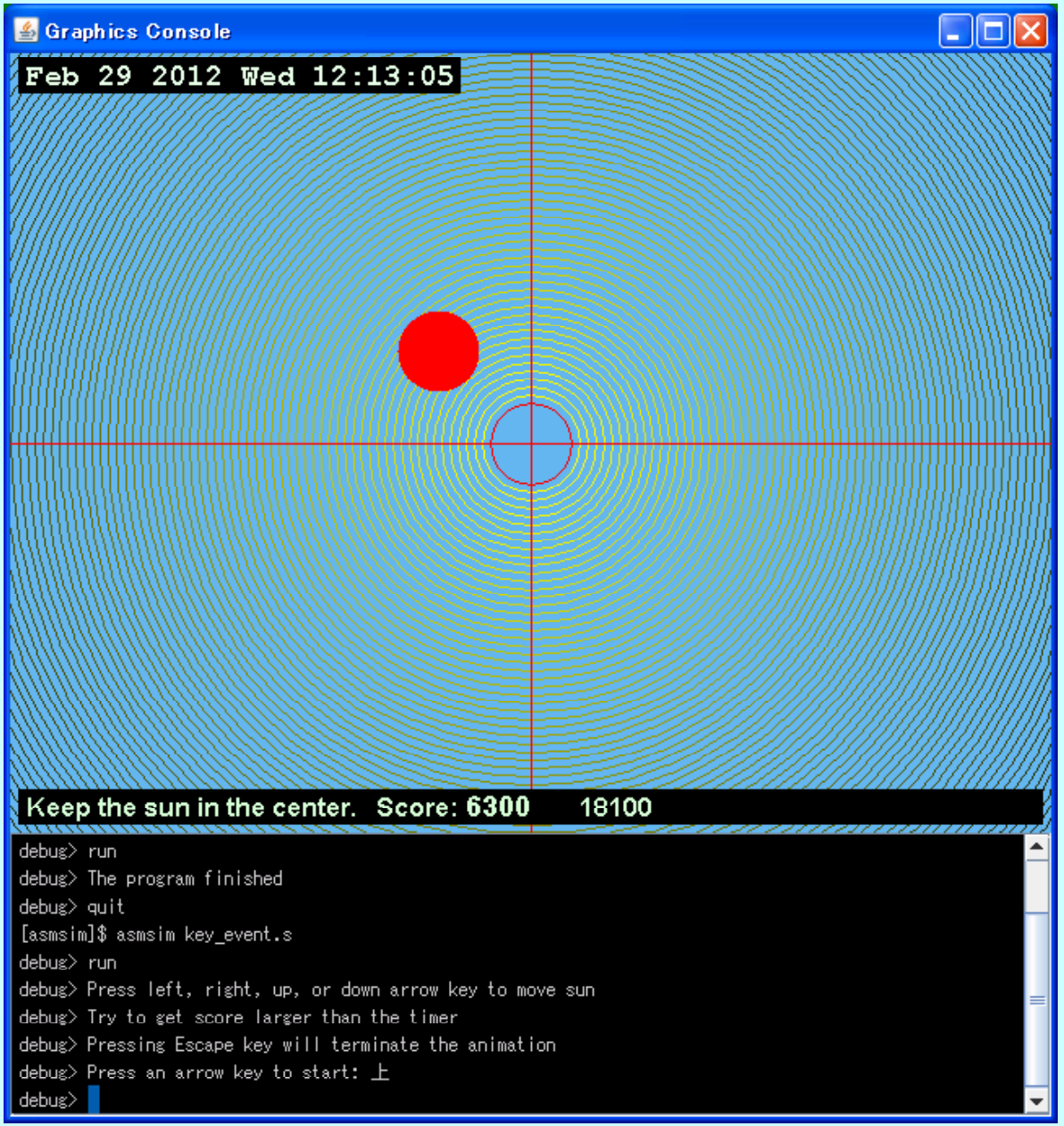


The Graphics Output Example (fonts.s)

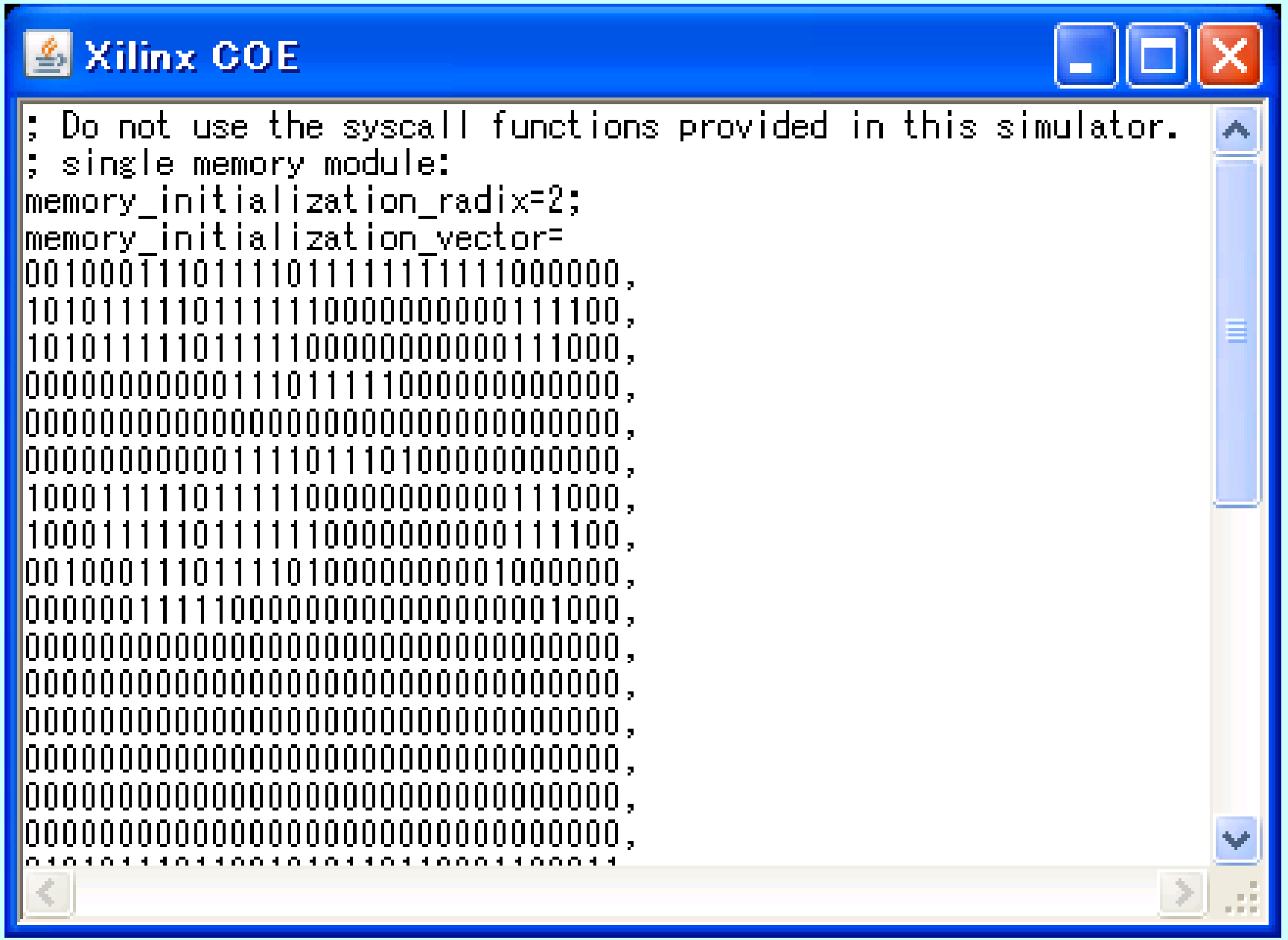
The Graphics Output Example (kanji.s)



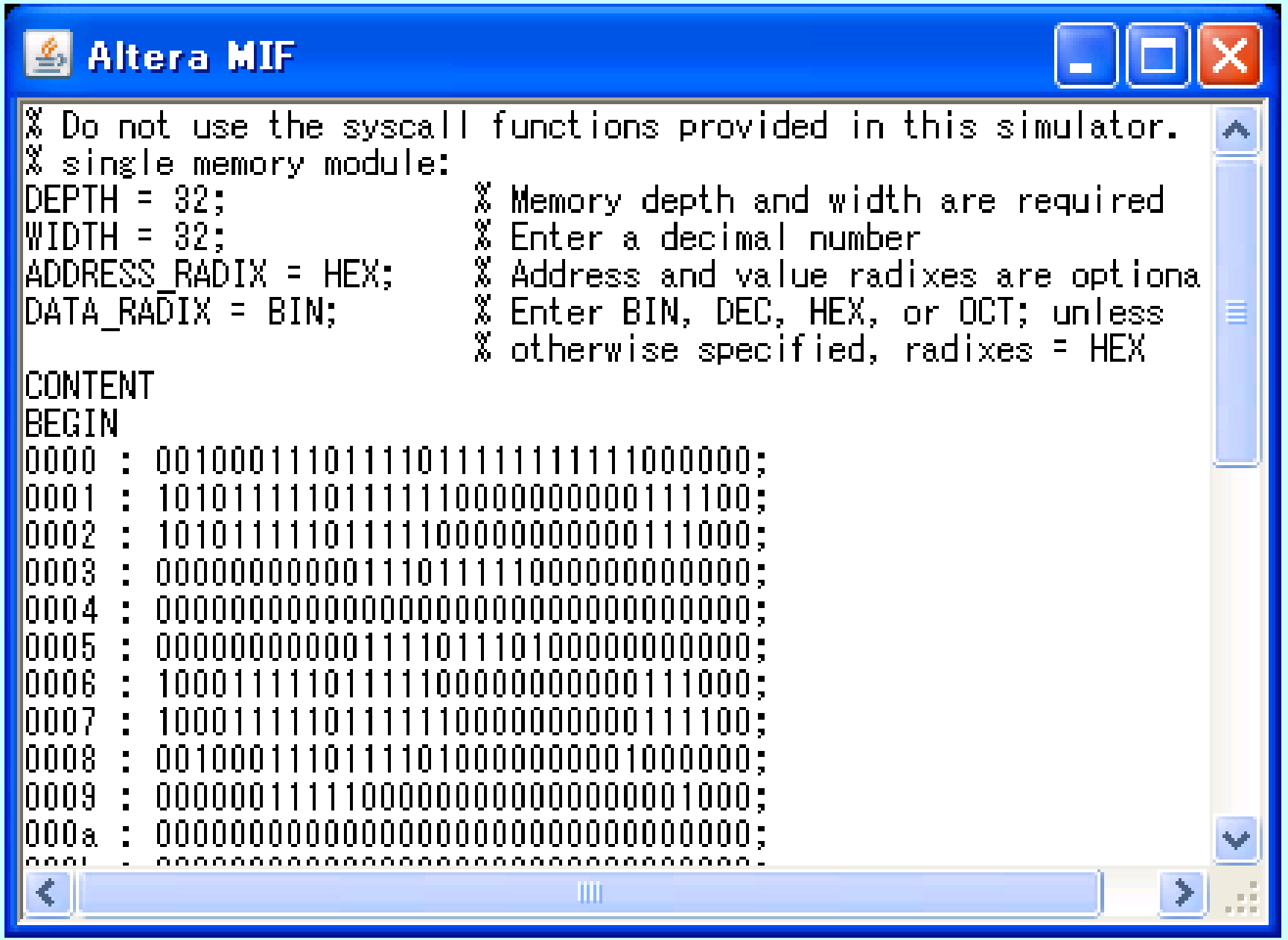
The Graphics Output Example (key_event.s)



Xilinx COE Window

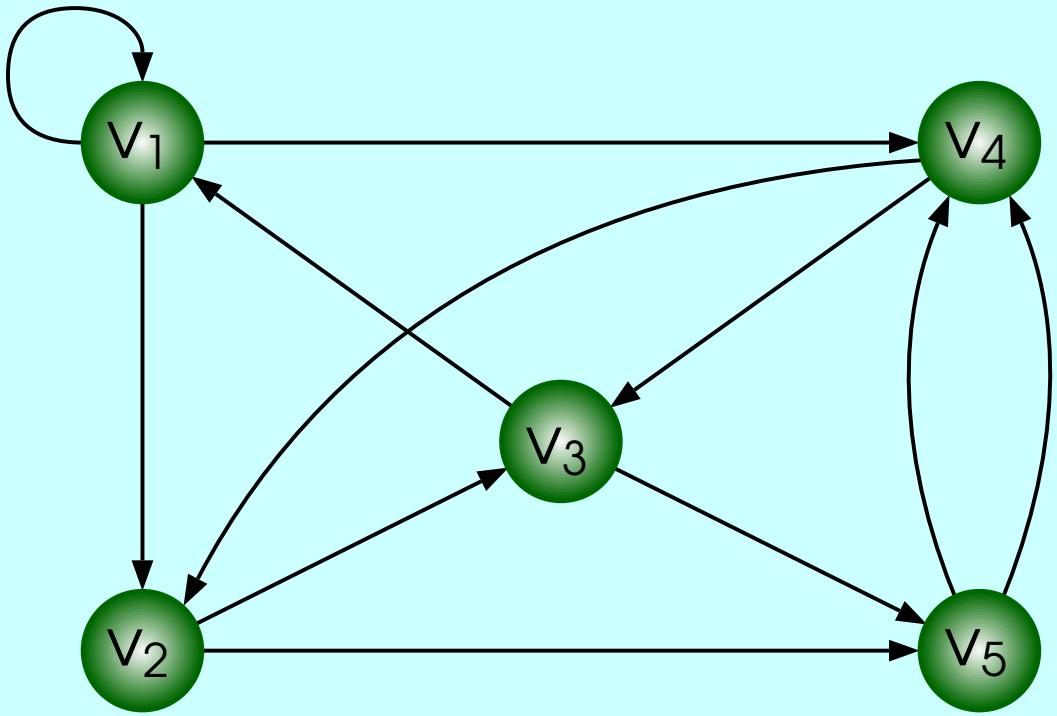


Altera MIF Window

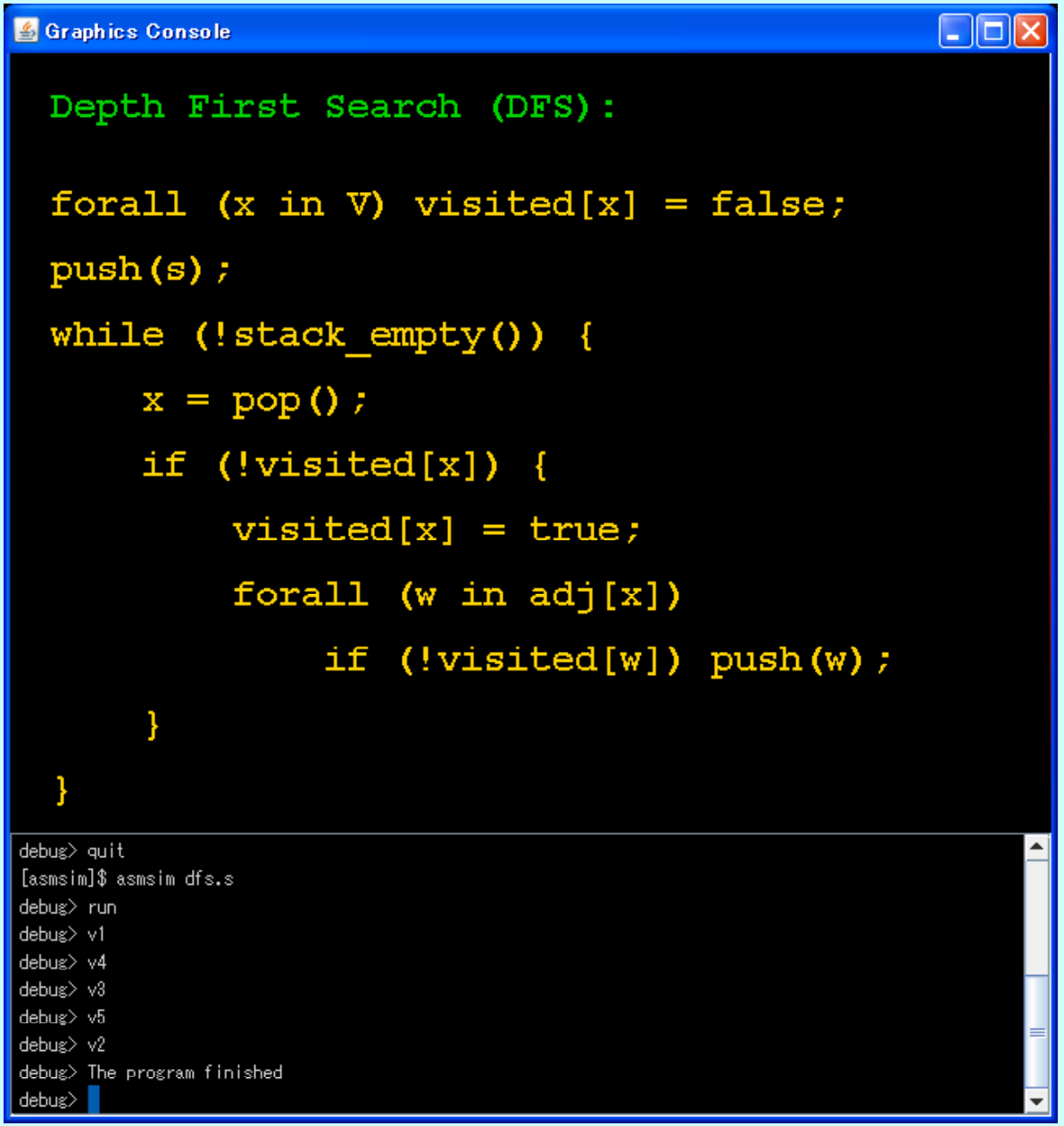


Support for Linked-List Data Structure

```
`define NULL 0x0
.data
v1:   .word  1, p1
p1:   .word  v1, p2
p2:   .word  v2, p3
p3:   .word  v4, NULL
v2:   .word  2, p4
p4:   .word  v5, p5
p5:   .word  v3, NULL
v3:   .word  3, p6
p6:   .word  v1, p7
p7:   .word  v2, p8
p8:   .word  v5, NULL
v4:   .word  4, p9
p9:   .word  v5, p10
p10:  .word  v3, NULL
v5:   .word  5, p11
p11:  .word  v4, NULL
```



Depth First Search with Stack (dfs.s)



Breadth First Search with Queue (bfs.s)

Graphics Console

```
Breadth First Search (BFS):

forall (x in V) visited[x] = false;
enqueue(s);
while (!queue_empty()) {
    x = dequeue();
    if (!visited[x]) {
        visited[x] = true;
        forall (w in adj[x])
            if (!visited[w]) enqueue(w);
    }
}
```

```
debug> quit
[asmsim]$ asmsim bfs.s
debug> run
debug> v1
debug> v4
debug> v2
debug> v3
debug> v5
debug> The program finished
debug>
```

To Let Copy & Paste Work

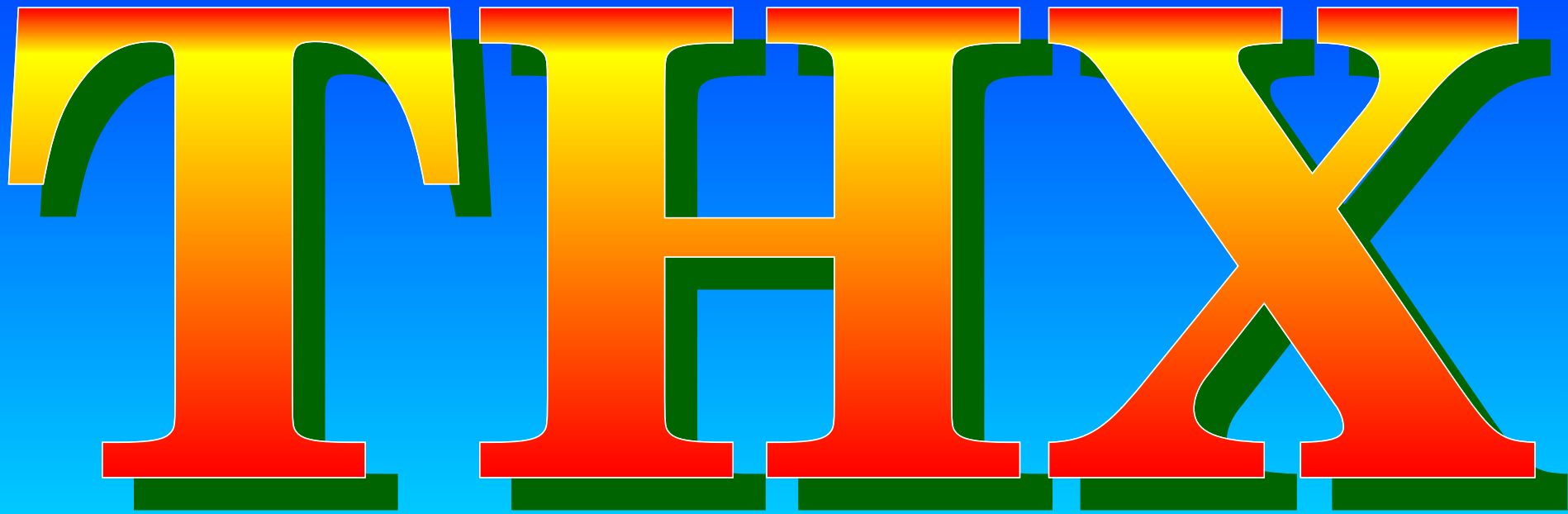
- Find "java.policy" file. It is usually located at C:\Program Files\Java\jre6\lib\security\
- Edit it: Add the following 3 lines in the top of the file:

```
grant codeBase "http://cis.k.hosei.ac.jp/-" {
      permission java.awt.AWTPermission "accessClipboard";
};
```

- Save it as ".java.policy" at your home directory (fold) (Note that this filename starts with a dot)
- Close the Web Broswer and open it again
- You should have Copy & Paste enabled now

Conclusions

- The assembly language programming is an important way to understanding how the computer works at the machine level
- AsmSim supports
  - System calls like printf() and scanf() in C
  - Graphics draws and fills
  - VRAM and VGA accesses
  - Keyboard interrupt mechanism
  - Linked list data structure
  - Automatic generation of the memory initialization files for Xilinx and Altera FPGAs
- AsmSim can be used by anyone anywhere anytime


THX