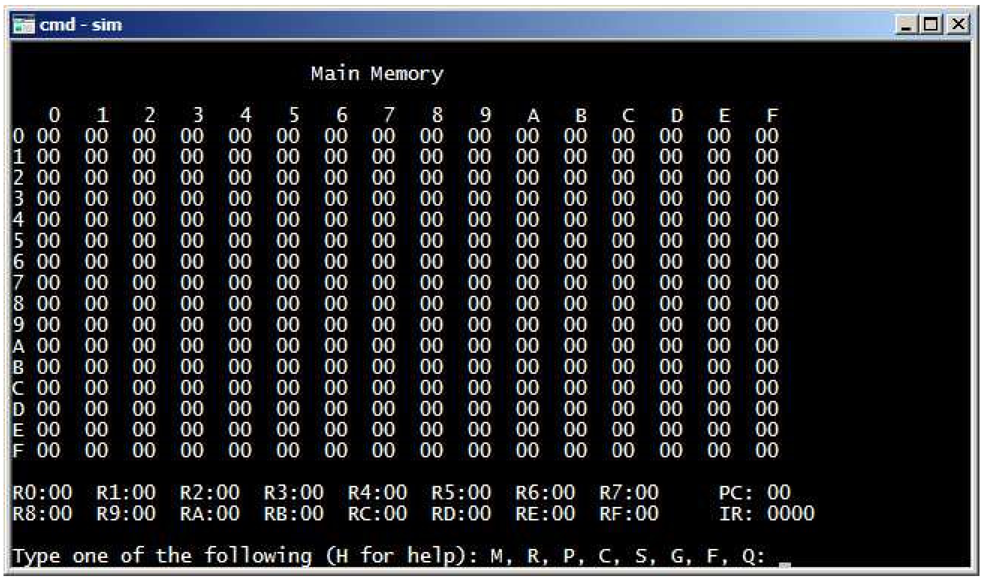
An Introduction to the Simple Computer Simulator

Chun-Jen Tsai 10/04/2006

The author of the textbook “Computer Science: An Overview” has designed a simple machine language for a hypothetical computer. He also wrote a Computer Simulator, sim.c, that can simulate the behavior of this hypothetical computer. The simulated computer has a Program Counter register, a Instruction register, 16 data registers, 256 main memory cells, and an ALU that can perform ADD, AND, OR, XOR, and ROR operations. The main memory cells are used to store both the machine program and data.

The simulator provided by the author has a command line user interface (see Figure 1), which is not so easy to use. On the other hand, the source code of the simulator is available to you so that you can modify it to make it become more powerful.

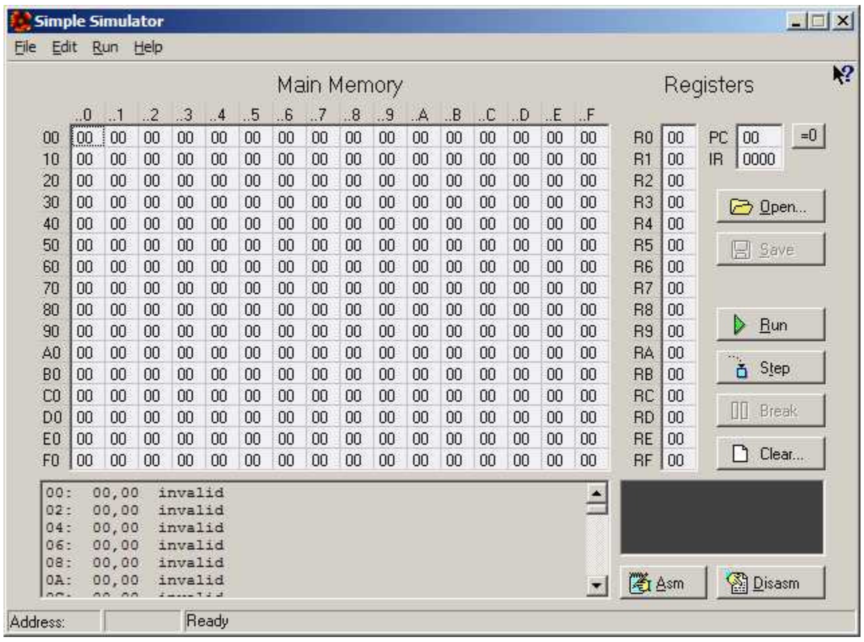


```
                        Main Memory

   0   1   2   3   4   5   6   7   8   9   A   B   C   D   E   F
0 00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
1 00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
2 00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
3 00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
4 00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
5 00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
6 00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
7 00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
8 00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
9 00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
A 00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
B 00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
C 00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
D 00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
E 00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
F 00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00

R0:00  R1:00  R2:00  R3:00  R4:00  R5:00  R6:00  R7:00     PC: 00
R8:00  R9:00  RA:00  RB:00  RC:00  RD:00  RE:00  RF:00     IR: 0000

Type one of the following (H for help): M, R, P, C, S, G, F, Q: _
```

Figure 1. The user interface of the command-line simulator

For our homework project, we will use a simulator with a nicer Graphical User Interface (GUI), called SimpSim. You can download the simulator from the following URL: http://wwwes.cs.utwente.nl/software/simpsim/index.html#download.

The program is a Windows program so you have to run it on a PC running Microsoft Windows XP Operating Systems. Upon execution, you will see the GUI of the simulator as shown in Figure 2.



	..0	..1	..2	..3	..4	..5	..6	..7	..8	..9	..A	..B	..C	..D	..E	..F
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
80	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
90	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

R0	00
R1	00
R2	00
R3	00
R4	00
R5	00
R6	00
R7	00
R8	00
R9	00
RA	00
RB	00
RC	00
RD	00
RE	00
RF	00

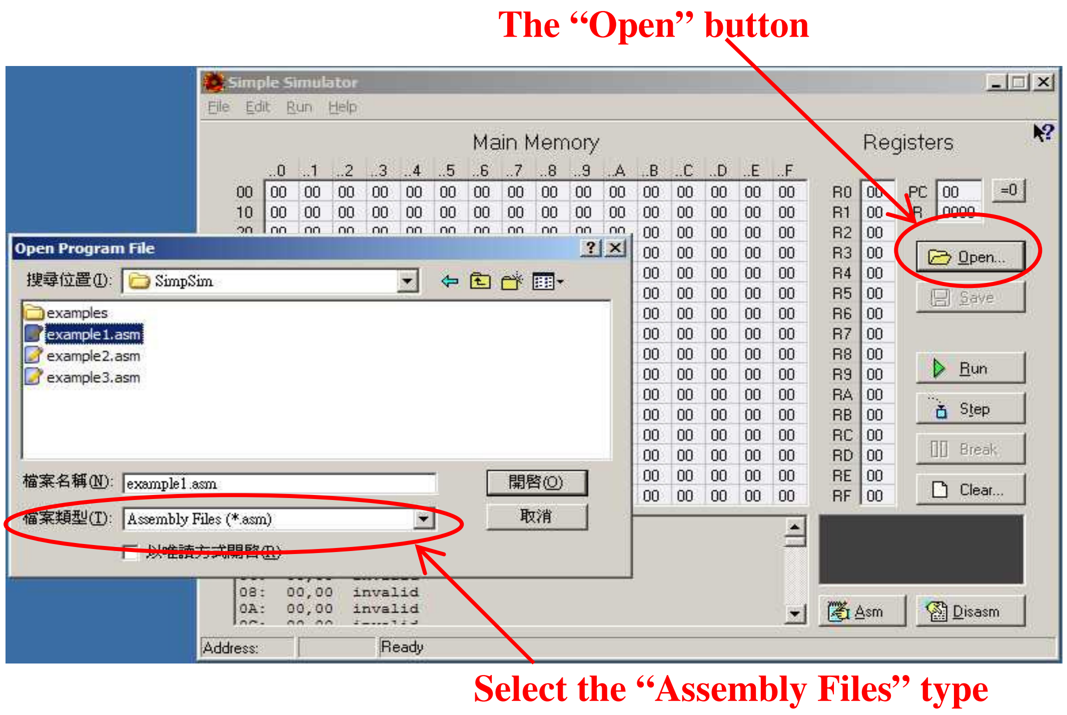
PC 00
IR 0000

```
00:  00,00  invalid
02:  00,00  invalid
04:  00,00  invalid
06:  00,00  invalid
08:  00,00  invalid
0A:  00,00  invalid
```

Figure 2. The GUI of the SimpSim simulator

Before you start using the simulator, you should recall that although a CPU only recognizes and executes the instruction code of a machine language, it is difficult for human to write a program in instruction code directly. Therefore, when we usually use a different form of the machine language called the assembly language.

You can use the “Open” button to load an assembly program (*.asm) or a machine code program (*.prg) into the simulator, as shown in Figure 3.



搜尋位置(I): SimpSim

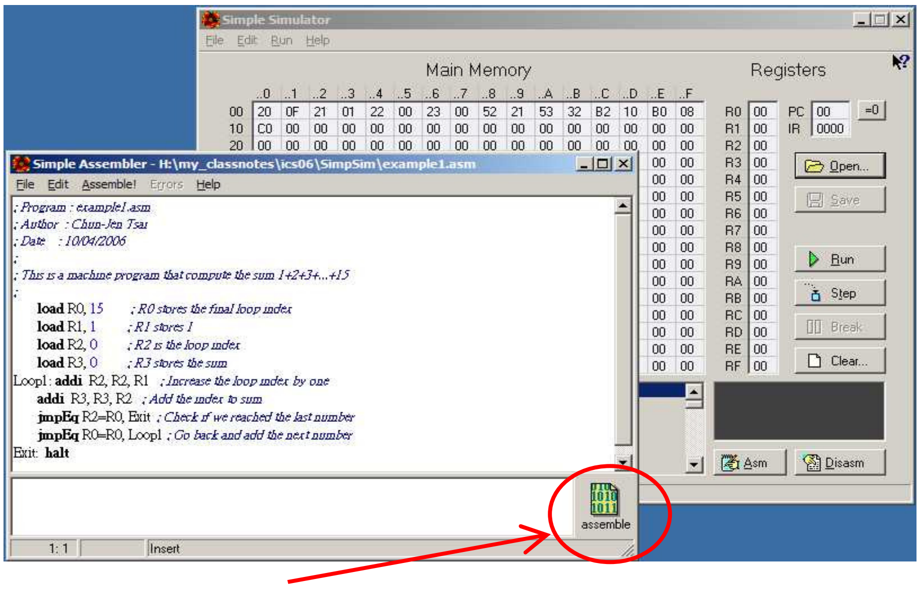
examples
example1.asm
example2.asm
example3.asm

檔案名稱(N): example1.asm
檔案類型(T): Assembly Files (*.asm)

```
08:  00,00  invalid
0A:  00,00  invalid
```

Figure 3. Load a program into the simulator

After loading the assembly program, you will see a window of the assembly source code. You can press the "assemble" button in the lower-right corner of the window to convert the assembly program into machine instruction code and load the machine code into the main memory, as shown in Figure 4.



```
    ..0 ..1 ..2 ..3 ..4 ..5 ..6 ..7 ..8 ..9 ..A ..B ..C ..D ..E ..F
00  20  0F  21  01  22  00  23  00  52  21  53  32  B2  10  B0  08
10  C0  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
20  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
```

```
R0 00   PC 00
R1 00   IR 0000
R2 00
R3 00
R4 00
R5 00
R6 00
R7 00
R8 00
R9 00
RA 00
RB 00
RC 00
RD 00
RE 00
RF 00
```

```
; Program : example1.asm
; Author  : Chun-Jen Tsai
; Date    : 10/04/2006
;
; This is a machine program that compute the sum 1+2+3+...+15
;
      load R0, 15       ; R0 stores the final loop index
      load R1, 1        ; R1 stores 1
      load R2, 0        ; R2 is the loop index
      load R3, 0        ; R3 stores the sum
Loop1: addi R2, R2, R1   ; Increase the loop index by one
      addi R3, R3, R2   ; Add the index to sum
      jmpEq R2=R0, Exit ; Check if we reached the last number
      jmpEq R0=R0, Loop1 ; Go back and add the next number
Exit: halt
```

Press the "assemble" button to generate machine code

Figure 4. Generate machine instruction code into main memory

Now, you can simulate the execution of the program by pressing the "Run" button or the "Step" button. The "Run" button causes the machine to execute the program until it hit the "halt" instruction (C000). The "Step" button only executes the next instruction pointed to by the Program Counter (PC) register, so that you can trace the execution of the program step-by-step.

Note that you can create a new program by pressing the "Asm" button. However, it seems that the text editor of the simulator has some problem under the Chinese version of Windows XP. If this is the case for you, you should try to use a stand-alone text editor, such as notepad, UltraEdit, or PsPad, to edit your program.

In order to be familiar with the machine language of the simple computer, you should read chapter 2, section 8.6 of chapter 8, and the on-line Help menu of the SimpSim program carefully. In particular, section 8.6 of chapter 8 explains the machine instruction with op-code "D" and "E" in great detail. A summary of all the instructions supported by the SimpSim simulator is listed in the next page.

A **Summary of the Machine Instructions of the SimpSim Simulator**

The following table lists all the instructions of the simple machine language mentioned in the textbook. Please note that, in the table, XY stands for an 8-bit bit pattern, and X stands for a 4-bit bit pattern. Finally, R, S, and T stand for registers (such as R0, R1, R2, etc.).

The third column of the table (Assembly Instruction) is a shorthand notation for the machine instruction (sometimes we call this mnemonics representation of the machine language).

Machine Instruction		Assembly Instruction	Operation
Op-code	Operand		
1	RXY	load R, [XY]	Load R with the content from the memory cell at address XY
2	RXY	load R, XY	Load R with the bit pattern XY
3	RXY	store R, [XY]	Store the content of R into the memory cell at address XY
4	0RS	move S, R	Move content of R into S
5	RST	addi R, S, T	Add S and T and put the result in R (R, S, and T are in two's complement integer notation)
6	ADD	addf R, S, T	Add S and T and put the result in R (R, S, and T are in floating-point notation)
7	RST	or R, S, T	OR the bit patterns in S and T and put the result in R
8	RST	and R, S, T	AND the bit patterns in S and T and put the result in R
9	RST	xor R, S, T	XOR the bit patterns in S and T and put the result in R
A	R0X	ror R, X	Circularly rotate the bit pattern in R one bit to the right X times
B	RXY	jmpEQ R=R0, XY	Start decoding the instruction located at address XY if the bit pattern in R is equal to the bit pattern in register 0
C	000	halt	Halt execution
D	0RS	load R, [S]	Load R with the content from the memory cell whose address is in S
E	0RS	store R, [S]	Store the content of R into the memory cell whose address is in S
F	RXY	jmpLE R<=R0, XY	Start decoding the instruction located at address XY if the bit pattern in R is less than or equal to the bit pattern in register 0