

p88 Assembler & Simulator (Version. 1.02, Mar. 2007)

What is p88 Assembler & Simulator ?

P88 is a very simple assembly language. It is originally described in "*Great Ideas in Computer Science*" (Alan W. Biermann, The MIT Press, 1990), which is a textbook for the beginners of computer science. P88 is used to explain assembly languages, computer architecture, and compilers.

The main target of p88 assembler & simulator is to provide compact assembly programming environment to novices, especially students of computer science courses. Because the specification of the language should be understood easily in the classroom, complex and detailed function is not needed. Though this system enhanced some functions for programming, the scale of the implementation would be suppressed in an adequate level.

How to Use

```
./p88 [options] source
Options:
  -t    trace
  -c    Syntax check only
  -d    Dump machine code
  -h    Show this help
```

Simple example

Make a source file using some text editor as the following. Except the last line, the first character should be a space or a tab. The source file should be plain text. The extension of the file name is arbitrary. In this example, suppose that the file name is "ex.p88".

```
        IN      AX
        IN      BX
        CMP     AX, BX
        JB     LAB
        COPY    AX, BX
LAB     OUT     AX
```

If p88 simulator named "p88" (or "p88.exe") exists in the current directory, you can execute the program with the following command ("% " is a prompt character). This program reads two numbers from the terminal and prints the smaller one.

```
% ./p88 ex.p88
in ? 67
in ? 43
out: 43
>>> Program terminated. (Halt)
```

Notations

Each line of source program

[label]	Lines with a label only, or empty lines are allowed.
[label] operation	One operation is written in a line. A label can be added.
[label] directive	One directive written in a line. A label can be added.
; comment	“;” shows the beginning of comment. Comment can be appended at the end of command or directive.

Each label should be written from the first column of the line. Without label, spaces (SP or TAB) should be placed ahead to the operation or the directive.

Words written in program

In program, the case of letters is ignored. A label is a string of alphabets or digits, and the first character should be an alphabet. P88 simulator recognizes first 7 characters only. For example, "FUNCTION", "FUNCTION2", and "FUNCTIOO" are recognized as the same label.

You can not use opcodes, register names, and other reserved words as labels.

Reserved Words

COPY, ADD, SUB, MUL, DIV, CMP, JMP, JNB, JB, IN, OUT,
CALL, RTN, PUSH, POP, END, AX, BX, CX, DX, RES, RANDOM

Numbers

P88 simulator can handle only 16 bits signed integers (from -32768 to +32767). Numbers written in program should in decimal format. Immediate values in program are 8 bits signed integers (from -128 to 127). You must prefix immediate values with "#".

Operators and Operands

Explanatory notes:

AX, BX, CX, DX registers
mem memory address (label or integer in decimal)
#8 immediate value: an integer prefixed with "#" (from -128 to 127)

END	Halt the program. No operand is needed.
COPY AX,BX	Copy the contents of register BX into register AX.
COPY AX,mem	Copy the contents of memory mem into register AX.
COPY AX,c(BX)	Copy the contents of the memory whose address is the contents of register BX into register AX.
COPY AX,mem+c(BX)	Copy the contents of the memory whose address is the sum of mem and the contents of register BX into register AX. (Note: 0+c(BX) is equals to c(BX). If BX==0, mem+c(BX) is equals to mem.)
COPY AX,#8	Copy immediately the 8 bits signed integer into register AX.
COPY mem,AX	Copy the contents of register AX into memory mem.
COPY c(BX),AX	Copy the contents of register AX into the memory whose address is the contents of register BX.
COPY mem+c(BX),AX	Copy the contents of register AX into the memory whose address is the sum of mem and the contents of register BX.
ADD AX,BX	Add the contents of register AX and BX. The result remains in register AX.
ADD AX,mem	Add the contents of register AX and the contents of memory mem. The result remains in register AX.
ADD AX,#8	Add immediately the contents of register AX and the 8 bits signed integer. The result remains in register AX.
SUB, MUL, and DIV	They are operators of subtraction, multiplication, and division correspondingly. Their operands are written in the same manner as

ADD operator. Note that the division by 0 causes an execution error.

CMP AX,BX	Compare the contents of register AX and BX.
CMP AX,mem	Compare the contents of register AX and the contents of memory mem.
CMP AX,#8	Compare immediately the contents of register AX and the 8 bits signed integer. CMP operation changes CF (Condition Flag). If AX is smaller, CF becomes B (Below). If AX is greater or equals to the second operand, CF becomes NB (Not Below).
JMP mem	Jump to the address mem without concerning CF.
JNB mem	Jump to the address mem, if CF is NB.
JB mem	Jump to the address mem, if CF is B.
CALL mem	Decrement the stack pointer (register DX) by two. Then, copy the return address, that is, the current instruction pointer added by two to the stack top, and jump to the address mem.
RTN	Get an address from the stack, and increment the stack pointer (register DX) by two. Then, jump to the address.
PUSH AX	Decrement the stack pointer (register DX) by two. Then, copy the contents of register AX to the stack top.
POP AX	Copy the contents of the stack top to register AX. Then, increment the stack pointer (register DX) by two.
IN AX	Read a 16 bits signed integer into register AX from the terminal.
OUT AX	Display the contents of register AX on the terminal.

Directives

n	Reserve 2 bytes memory with value n. You can specify a label or an integer as n.
RES n	Reserve n bytes memory.

Behavior of p88 Simulator

Addressing

Only byte-addressing is supported by p88 assembler & simulator. On the other hand, p88 simulator can deal word (2 bytes) data only. COPY operation, arithmetic operations, and comparison operation are only for word data. However, you do not have to align data to word boundary. Though the address space of p88 simulator is only 4096 bytes (12 bits), it would be enough to execute tiny programs.

Stack related operations (PUSH, POP, CALL, and RTN) are newly added. Though they are not described in the original text, they are needed to realize subroutine call. Register DX is used as the stack pointer, and the initial value of register DX is set to be 4094.

Execution

Program always starts from address 0, and terminates by execution of END operation. Division by zero, execution of illegal operation, or interrupts from the terminal also terminates program. Program can access any memory areas regardless of codes or data.

In case that program has fallen into infinite loop, interrupt (type control-C) from the terminal to quit.

Special features

If the last operation of the program is not END, p88 assembler automatically adds END operation. Additionally, if the program has undefined labels, p88 assembler automatically adds 2 bytes area each to define labels. Such special behavior is set up to run sample programs described in Biermann's original text. Program with undefined labels is not legal, of course, so undefined labels are explicitly displayed before the execution.

Memory location referred to as RANDOM is available, which is described in Biermann's original text. Loading from RANDOM always returns different random number (from 0 to 32767).

Trace mode

If command line option -t is specified, the contents of registers are displayed for each operation (trace mode). Option -d shows the machine code (intermediate code). The following is an example. The contents of registers (AX, BX, CX, DX) and CF are displayed at the right of the screen. Note that the values are displayed just before the execution of the corresponding operation. Addresses in machine code are displayed with the notation such as "[23]" instead of labels.

```
% ./p88 -d -t t08b.p88
----- MEMORY DUMP -----
10 00 e1 1d c0 17 20 02 60 0a 90 02 30 02 1d 80
17 f1 60 01 80 0c 00 00 00 00 00 00 00 00 00
00
-----
. . . . . 0: COPY AX,#0 0 0 0 4094 NB
. . . . . 2: IN BX 0 0 0 4094 NB
in ? 4
. . . . . 3: COPY [23]+c(AX),BX 0 4 0 4094 NB
. . . . . 6: ADD AX,#2 0 4 0 4094 NB
. . . . . 8: CMP AX,#10 2 4 0 4094 NB
. . . . . 10: JB [2] 2 4 0 4094 B
. . . . . 2: IN BX 2 4 0 4094 B
```

Copyright

Copyright of this software (including source programs and documents) belongs to the author, Takeshi Ogihara.

This software is freeware. You may freely copy and redistribute it. Permission is granted to modify the source for your own purposes. But DON'T redistribute modifications without this copyright notice and a declaration of modification.

There is absolutely no warrantee on this software. The author takes no responsibility for any damage caused by this software.

Send suggestions and bug reports to: t_ogihara@mac.com

(C) 2006, 2007, OGIHARA Takeshi