Lab 6a: LC-3 Assembler

Write a program which can turn LC-3 assembly code to binary machine code.

Implementation Details

- You are required to write in **C** or any other high level programming language.

Instructions

The instructions you need to implement is listed in the following table.

	Assembly	Example	Binary
ADD	`ADD reg, reg, reg` `ADD reg, reg, imm5`	`ADD R1, R2, R3` `ADD R1, R2, #3`	`0001 001 010 000 011` `0001 001 010 1 00011`
AND	`AND reg, reg, reg` `AND reg, reg, imm5`	`AND R1, R2, R3` `AND R1, R2, #3`	`0101 001 010 000 011` `0101 001 010 1 00011`
NOT	`NOT reg, reg`	`NOT R1, R2`	`1001 001 010 1 11111`
LD	`LD reg, off9`	`LD R1, A`	`0010 001 000000111`
LDR	`LDR reg, reg, imm6`	`LDR R1, R2, #3`	`0110 001 010 000011`
LDI	`LDI reg, off9`	`LDI R1, A`	`1010 001 000000111`
LEA	`LEA reg, off9`	`LEA R1, A`	`1110 001 000000111`
ST	`ST reg, off9`	`ST R1, A`	`0011 001 000000111`
STR	`STR reg, reg, imm6`	`STR R1, R2, #3`	`0111 001 010 000111`
STI	`STI reg, off9`	`STI R1, A`	`1011 001 000000111`
TRAP	`TRAP vect8` `trap`	`TRAP x25` `HALT`	`1111 0000 0010 0101` `1111 0000 0010 0101`
BR	`br* off9`	`BR A`	`0000 111 000000111`
JMP	`JMP reg` `RET`	`JMP R1` `RET`	`1100 000 001 000000` `1100 000 111 000000`
JSR	`JSR off11` `JSRR reg`	`JSR A` `JSRR R1`	`0100 1 00000000011` `0100 000 001 000000`
RTI	`RTI`	`RTI`	`1000 0000 0000 0000`

	Assembly	Example	Binary
.ORIG	`.ORIG vect16`	`.ORIG x3000`	`0011 0000 0000 0000`
.FILL	`.FILL imm16`	`.FILL #3`	`0000 0000 0000 0011`
.BLKW	`.BLKW #` unsigned decimal number	`.BLKW #3`	`0111 0111 0111 0111` `0111 0111 0111 0111` `0111 0111 0111 0111`
.STRINGZ	`.STRINGZ "` string `"`	`.STRINGZ "A"`	`0000 0000 0100 0001` `0000 0000 0000 0000`
.END	`.END`	`.END`	

The lowercase words in the above table can be write in the following ways:

	Explaination	Examples	Note
`reg`	`R0` ~ `R7`	`R0`	
`imm*`	`#` decimal number `x` hexadecimal number	`#3` `x-3`	Can be negative number.
`off*`	Label `#` decimal number	`A` `#3`	In this lab, we do not use `x` hexadecimal number to represent offset.
`vect*`	`x` unsigned hexadecimal number	`x80`	
`br*`	`BR`, `BRn`, `BRz`, `BRp`, `BRnz`, `BRzp`, `BRnp`, `BRnzp`	`BR`	`BRzn`, `BRpz`, `BRpn` will not appear.
`trap`	`GETC`, `OUT`, `PUTS`, `IN`, `PUTSP`, `HALT`	`HALT`	

- The operands are always separated by a comma and a space character.
- `off*` can only be a label or a `#` decimal number. `vect*` can only be an `x` unsigned hexadecimal number.
- The operand of `.FILL` can only be an `imm16`. That is to say, `.FILL Label` will not appear.
- `.BLKW` always fills x7777 in the memory locations set aside.
- There are no escape characters in the string after `.STRINGZ`.
- You do not have to handle any situation not mentioned above.

Labels

- Labels only contain: letters `A-Z` `a-z`, numbers `0-9`, underlines `_`.
- Labels can start with: letters `A-Z` `a-z`, underlines `_`.
- Labels can not be instruction names or numbers.
- To make it easy, there are some situations that you do not have to deal with:
  - Labels do not occupy a separate line.
  - There is at most one label for one instruction.
  - There is no colon `:` after the label.

Comments

- Comments starting with `;` will not appear in this lab.

Input

The LC-3 assembly code is input from *stdin*. It always starts with `.ORIG` and ends with `.END`.

Sample 1

```
            .ORIG   x3000
            HALT
            .END
```

Sample 2

```
            .ORIG   x3000
            LD      R0, DATA
            AND     R2, R2, #0
            ADD     R3, R0, R0
            ADD     R4, R3, R3
            AND     R1, R3, R4
            AND     R1, R1, R0
            BRz     NO
            ADD     R2, R2, #1
NO          HALT
DATA        .FILL   x1234
            .END
```

- There may be empty lines in input.
- There is only one pair of `.ORIG` and `.END`.

Output

Print the LC-3 machine code to *stdout*.

Sample 1

```
0011000000000000
1111000000100101
```

Sample 2

```
0011000000000000
0010000000001000
0101010010100000
0001011000000000
0001100011000011
0101001011000100
0101001001000000
0000010000000001
0001010010100001
1111000000100101
0001001000110100
```

- The first line is the starting address of the program.
- Do not print any space characters.
- When testing, it is guaranteed that there is no error in the input assembly code. You do not need to do error handling and traceback.
- You can test your result by pasting the machine code into LC3Tools.