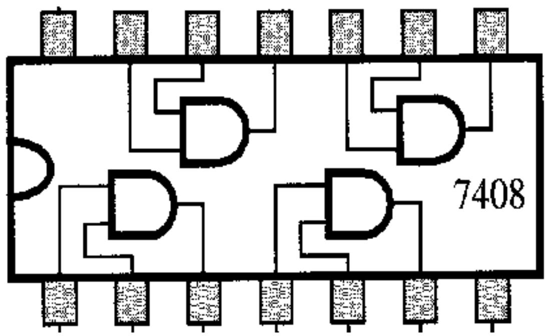
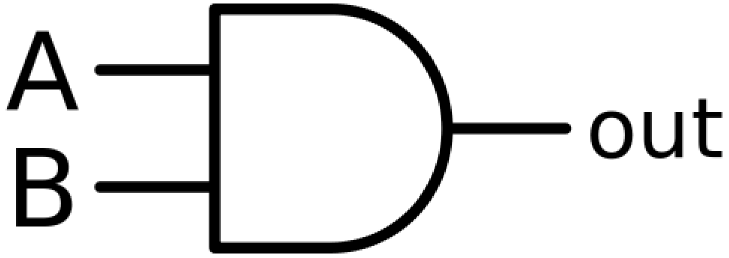
DIGITAL LOGIC

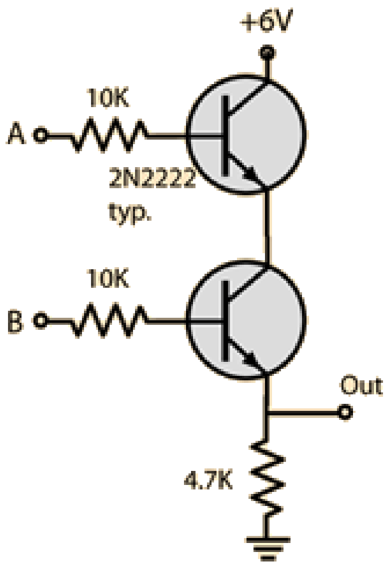
Assembly Digital Logic

Logic Gates and Boolean Expressions

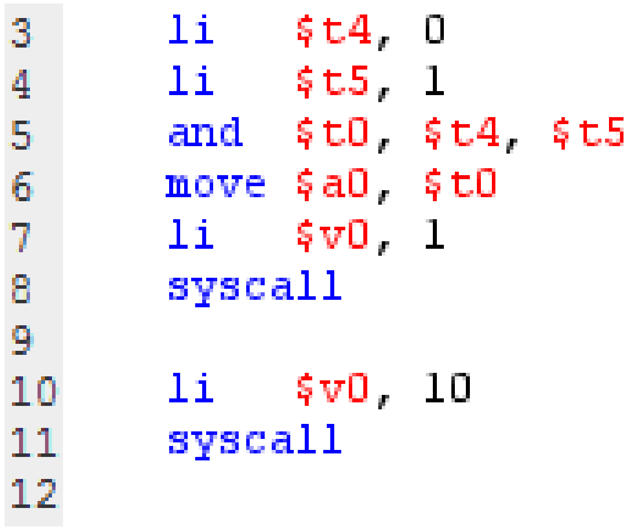
AND gate with 2-inputs





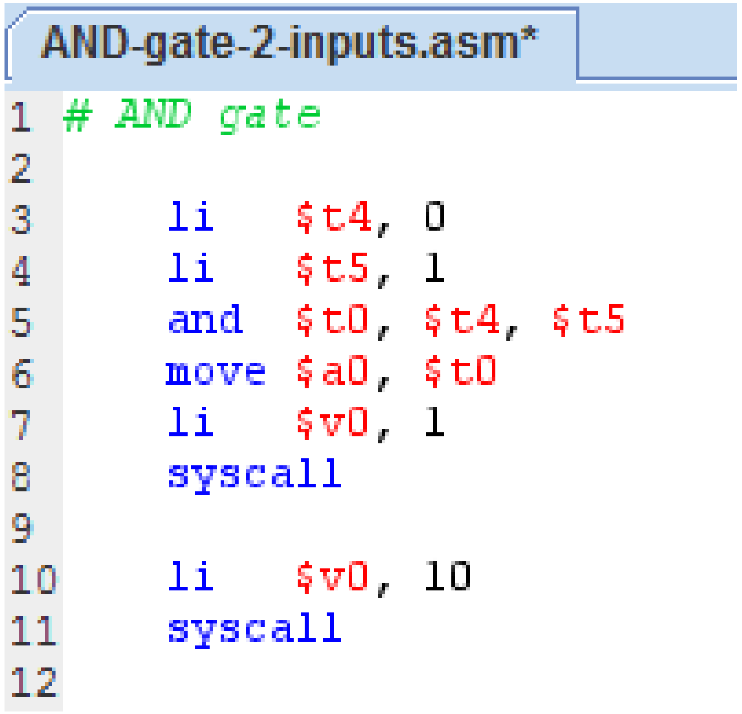


Demo-1



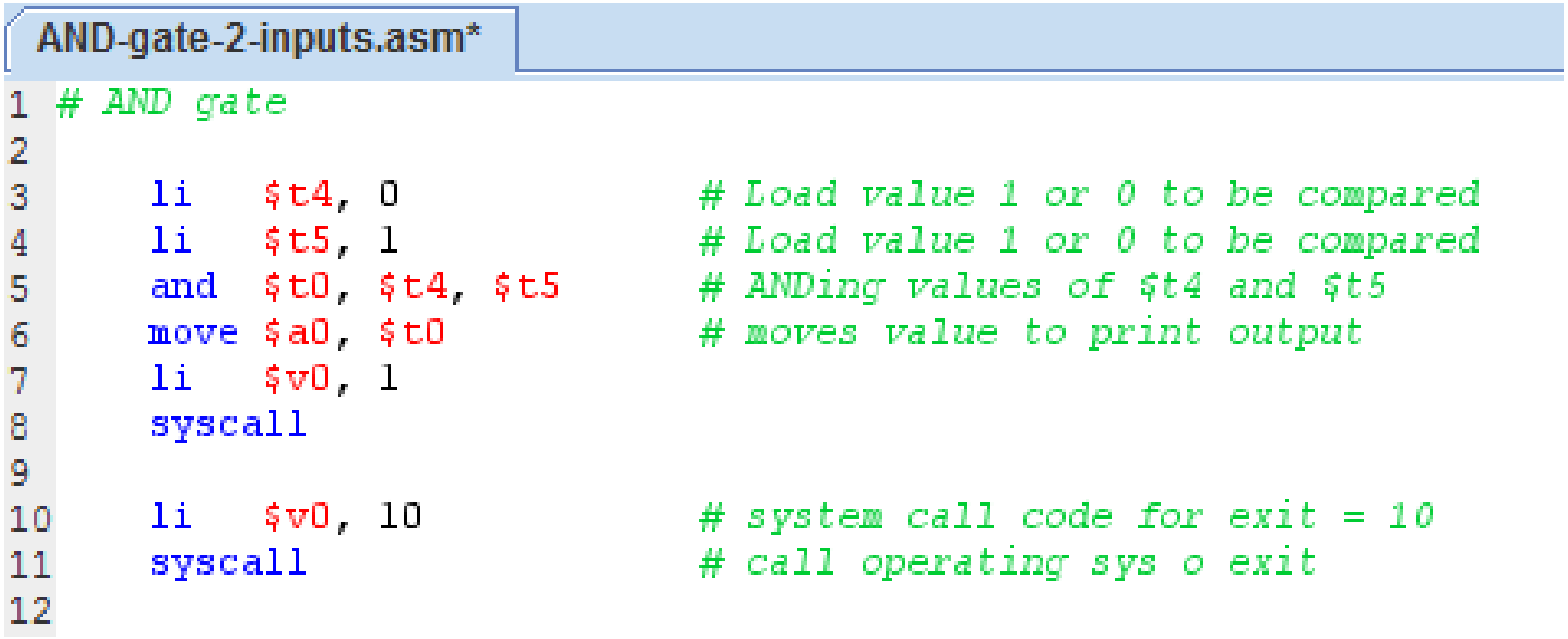
```
	li   $t4, 0
	li   $t5, 1
	and  $t0, $t4, $t5
	move $a0, $t0
	li   $v0, 1
	syscall

	li   $v0, 10
	syscall
```

AND gate with 2-inputs


AND-gate-2-inputs.asm*

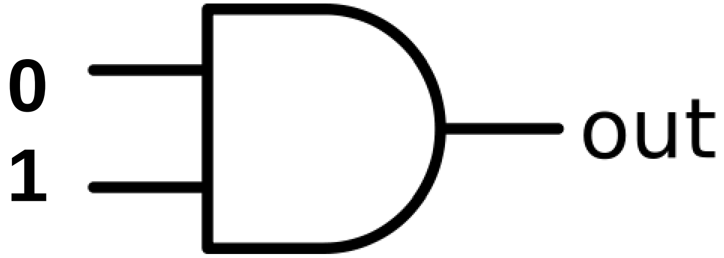
```
# AND gate

    li    $t4, 0
    li    $t5, 1
    and   $t0, $t4, $t5
    move  $a0, $t0
    li    $v0, 1
    syscall

    li    $v0, 10
    syscall
```

AND gate with 2-inputs

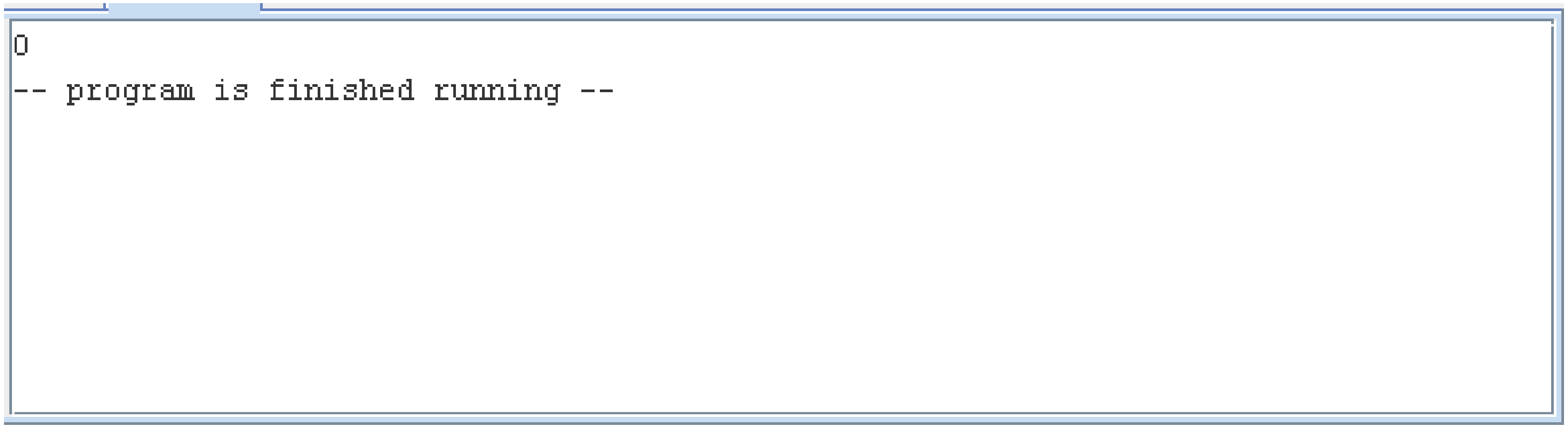


AND-gate-2-inputs.asm*

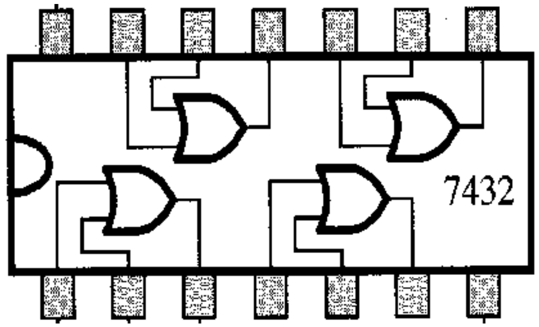
```
# AND gate

    li    $t4, 0              # Load value 1 or 0 to be compared
    li    $t5, 1              # Load value 1 or 0 to be compared
    and   $t0, $t4, $t5       # ANDing values of $t4 and $t5
    move  $a0, $t0            # moves value to print output
    li    $v0, 1
    syscall

    li    $v0, 10             # system call code for exit = 10
    syscall                   # call operating sys o exit
```

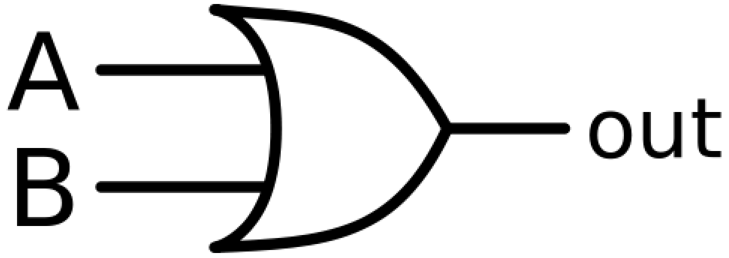


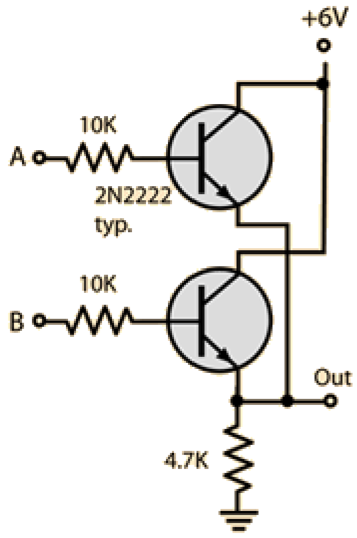
Assemble …GO


```
0
-- program is finished running --
```

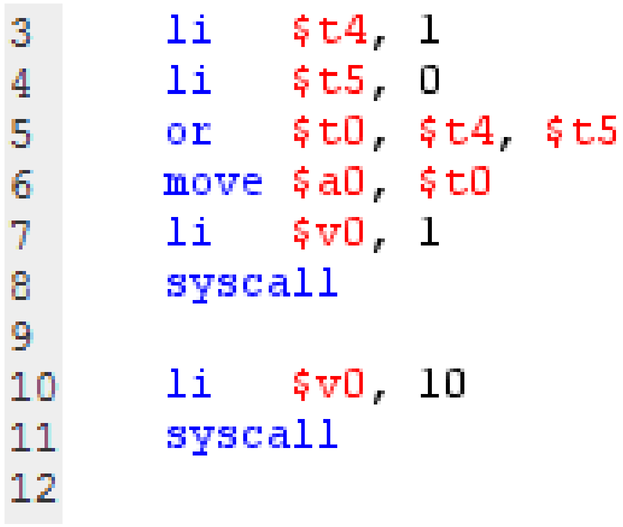
OR gate with 2-inputs





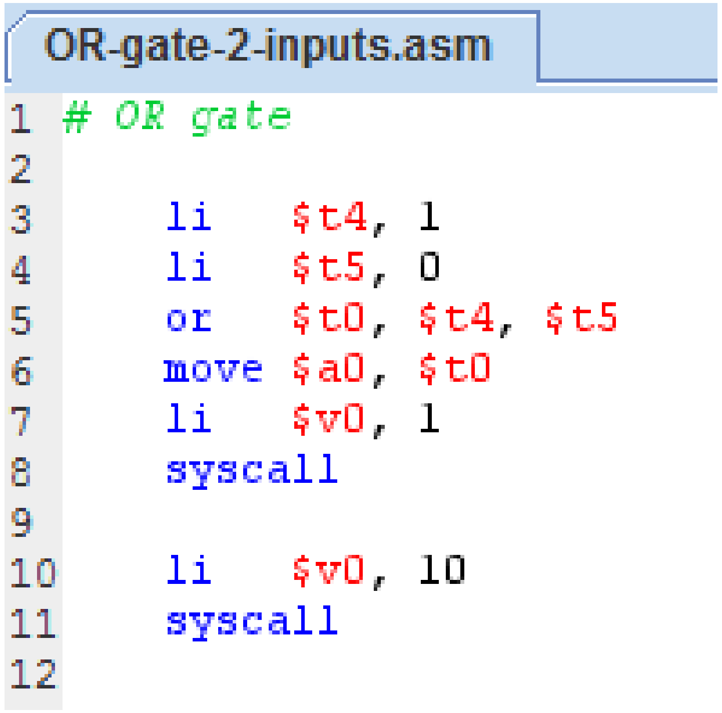


Demo-2



```
    li    $t4, 1
    li    $t5, 0
    or    $t0, $t4, $t5
    move  $a0, $t0
    li    $v0, 1
    syscall

    li    $v0, 10
    syscall

```

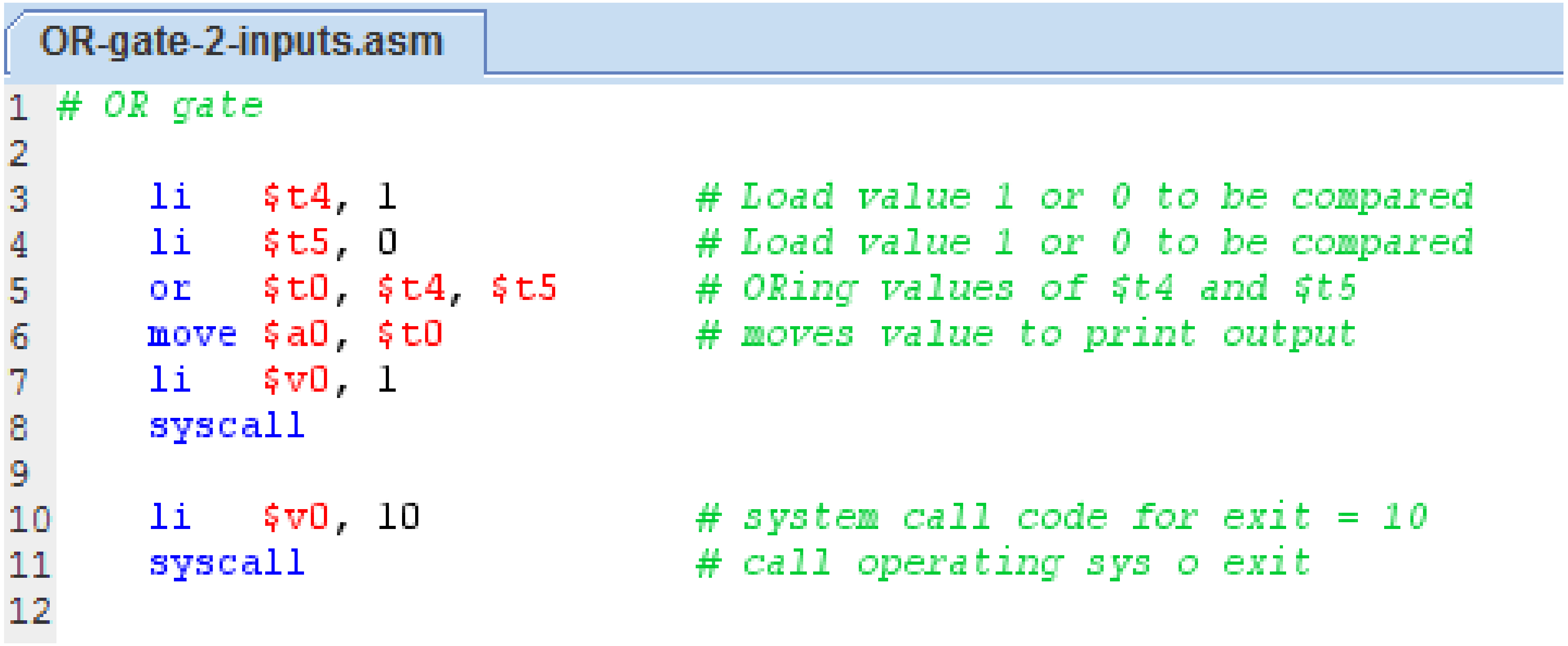
OR gate with 2-inputs



OR-gate-2-inputs.asm

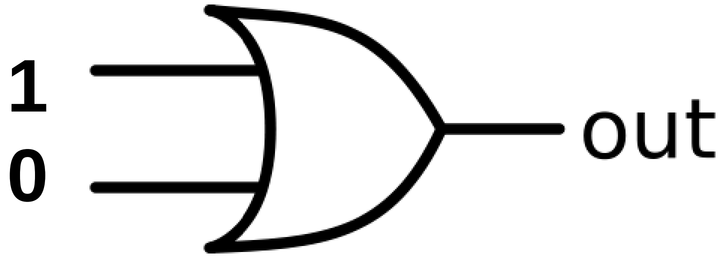
```
# OR gate

    li   $t4, 1
    li   $t5, 0
    or   $t0, $t4, $t5
    move $a0, $t0
    li   $v0, 1
    syscall

    li   $v0, 10
    syscall

```

OR gate with 2-inputs

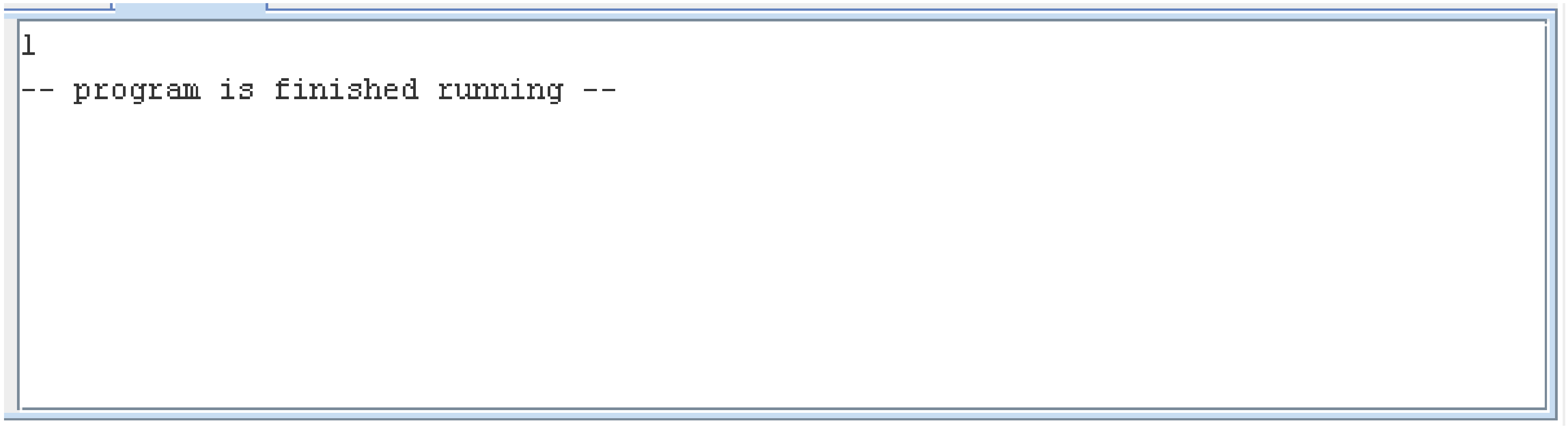


OR-gate-2-inputs.asm

```
# OR gate

    li    $t4, 1              # Load value 1 or 0 to be compared
    li    $t5, 0              # Load value 1 or 0 to be compared
    or    $t0, $t4, $t5       # ORing values of $t4 and $t5
    move $a0, $t0             # moves value to print output
    li    $v0, 1
    syscall

    li    $v0, 10             # system call code for exit = 10
    syscall                   # call operating sys o exit

```



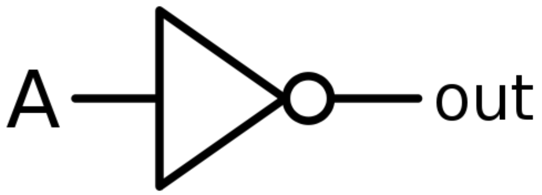
Assemble …GO

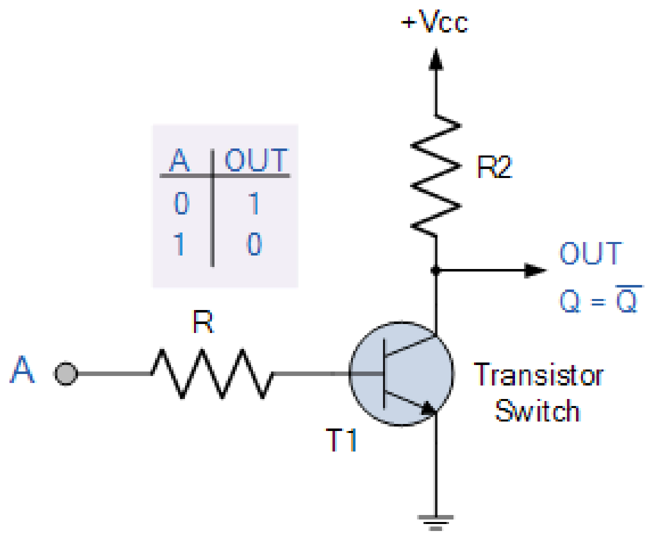


```
1
-- program is finished running --
```

Homework

- Write a simple Assembly Program for a **NOT** (Inverter) gate

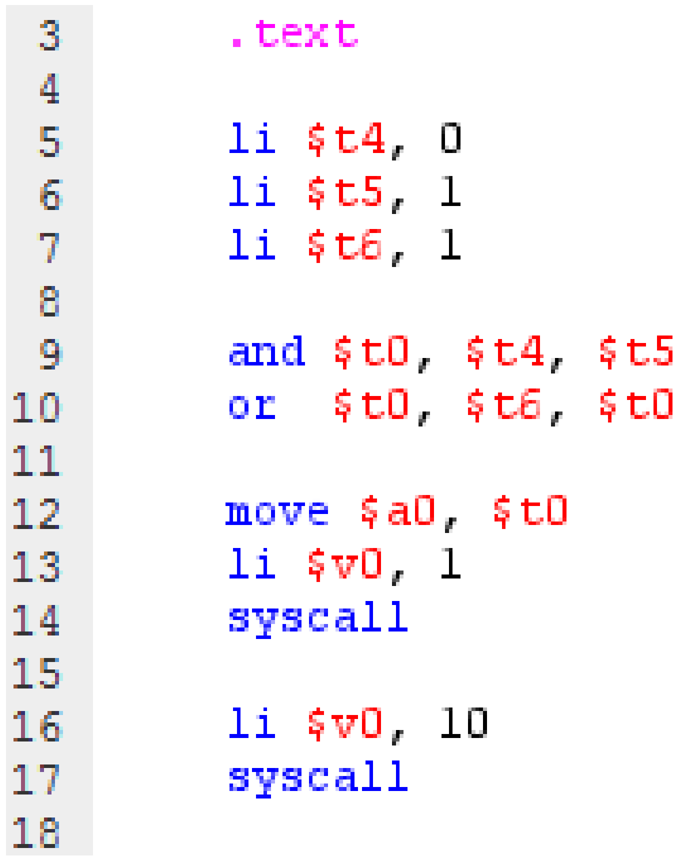




A	OUT
0	1
1	0

Digital Logic Expressions

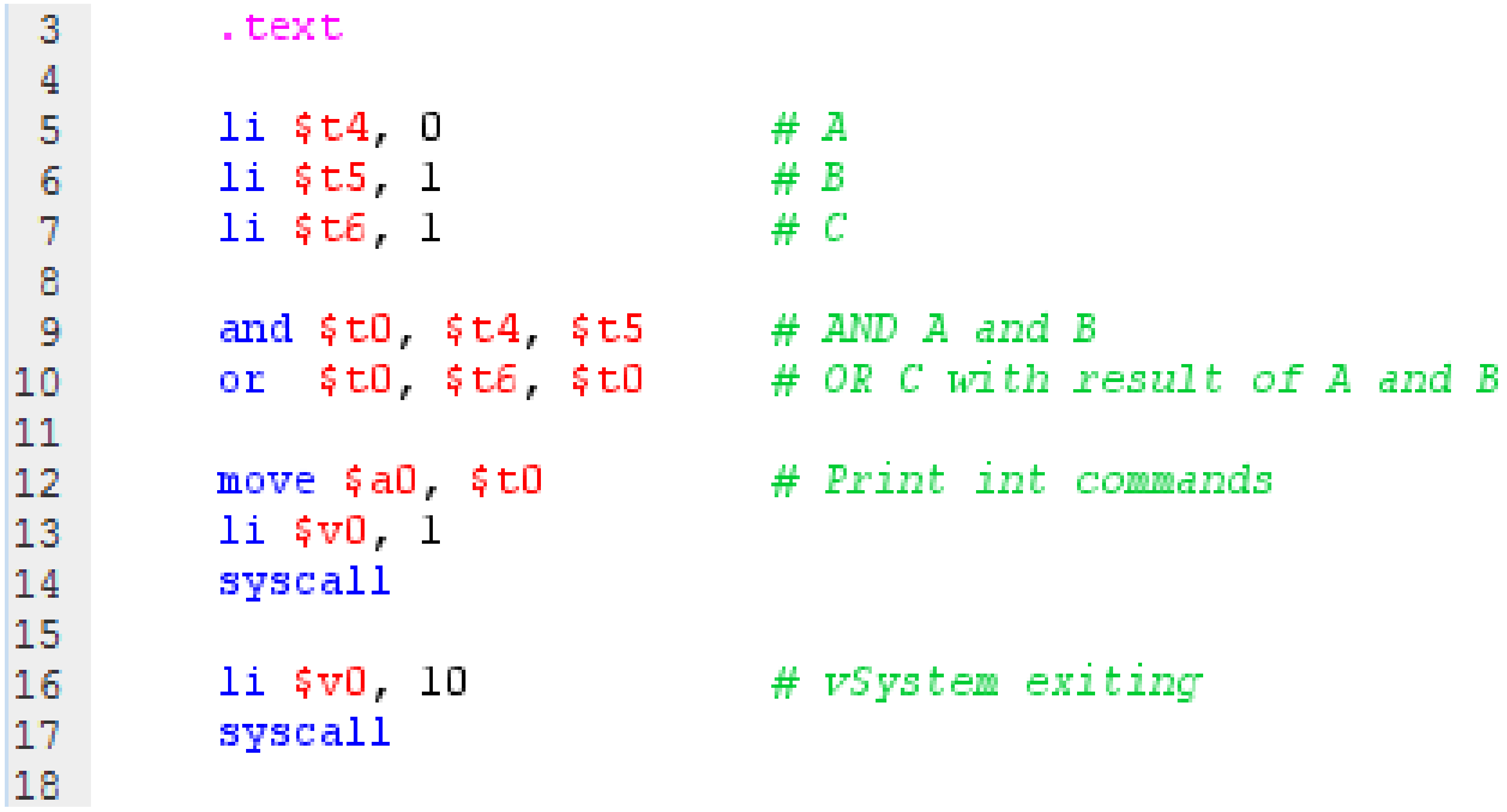
Logic (Assembly)



```
.text

li $t4, 0
li $t5, 1
li $t6, 1

and $t0, $t4, $t5
or  $t0, $t6, $t0

move $a0, $t0
li $v0, 1
syscall

li $v0, 10
syscall
```

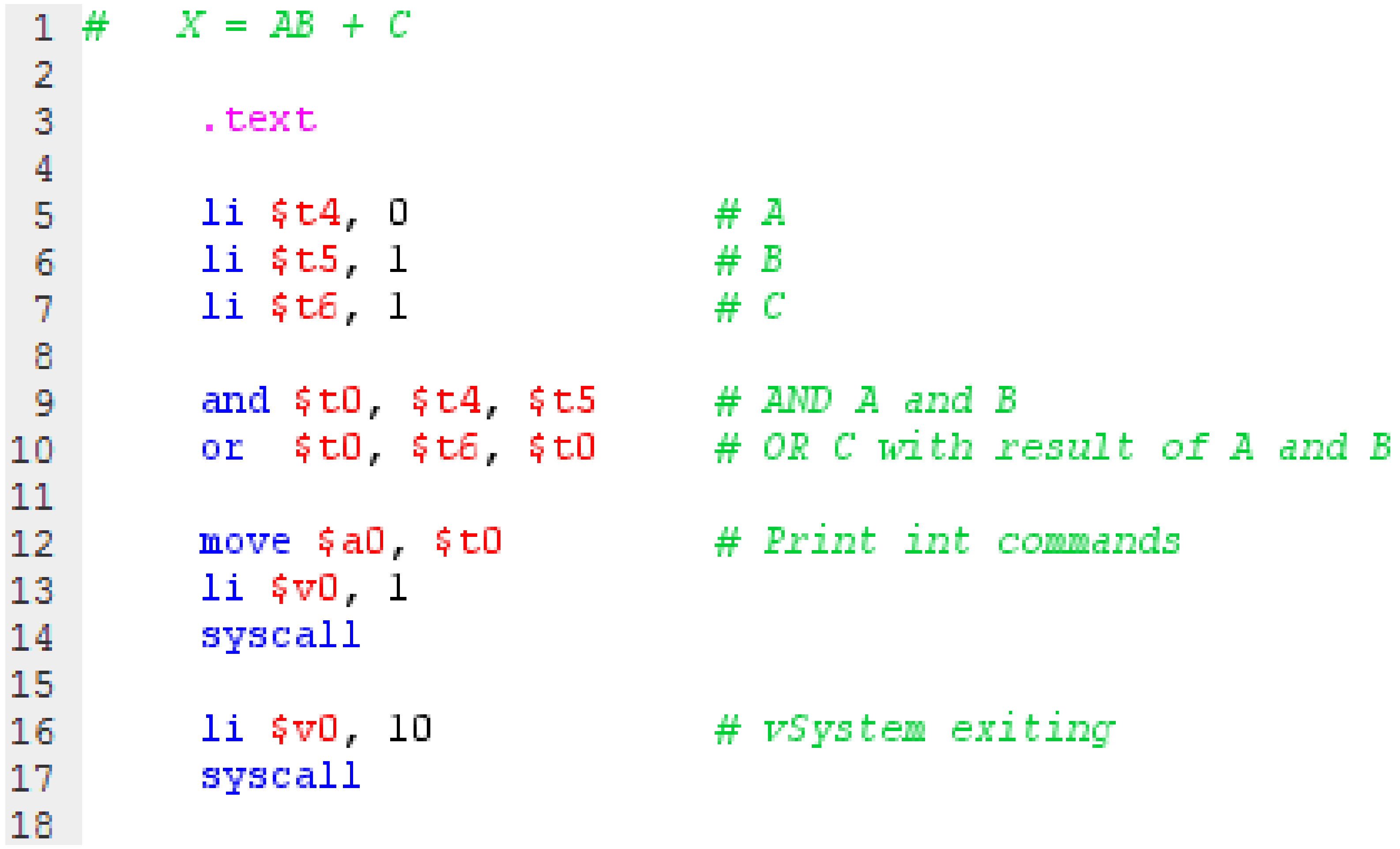
5 minutes to find out the function and the output of the demo-code

Logic (Assembly)



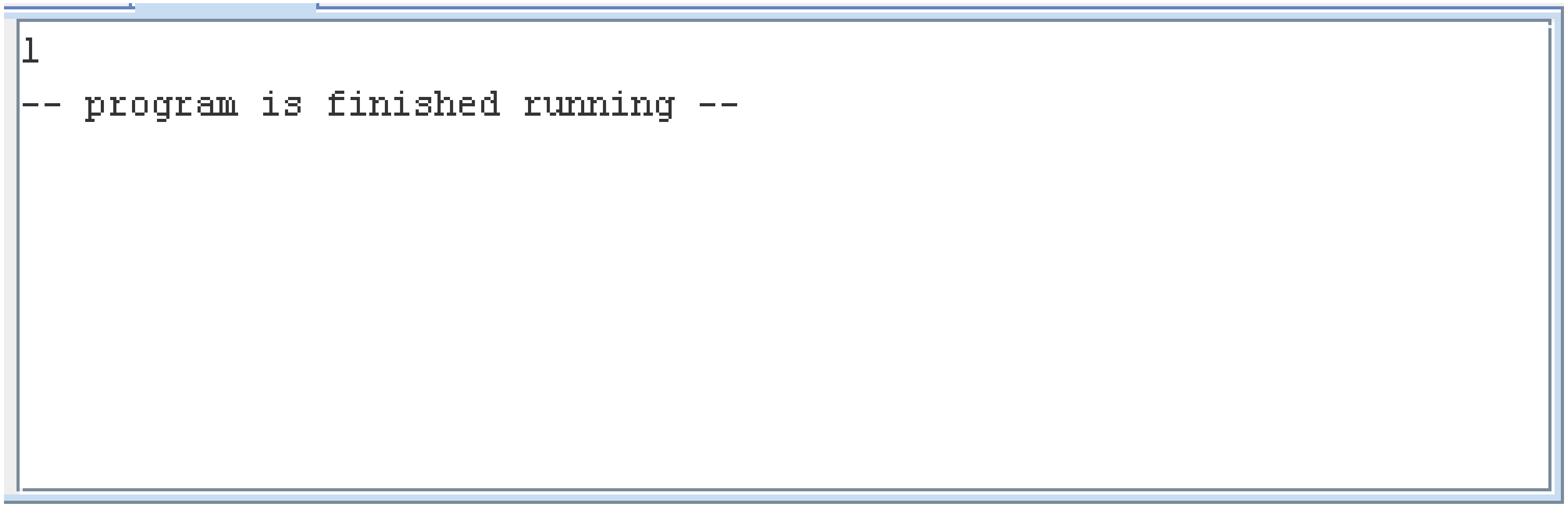
```
.text

li $t4, 0                   # A
li $t5, 1                   # B
li $t6, 1                   # C

and $t0, $t4, $t5           # AND A and B
or  $t0, $t6, $t0           # OR C with result of A and B

move $a0, $t0               # Print int commands
li $v0, 1
syscall

li $v0, 10                  # vSystem exiting
syscall
```

Logic Expression: $X = A \bullet B + C$



```
#    X = AB + C

        .text

        li $t4, 0                   # A
        li $t5, 1                   # B
        li $t6, 1                   # C

        and $t0, $t4, $t5           # AND A and B
        or  $t0, $t6, $t0           # OR C with result of A and B

        move $a0, $t0               # Print int commands
        li $v0, 1
        syscall

        li $v0, 10                  # vSystem exiting
        syscall

```

Assemble … GO



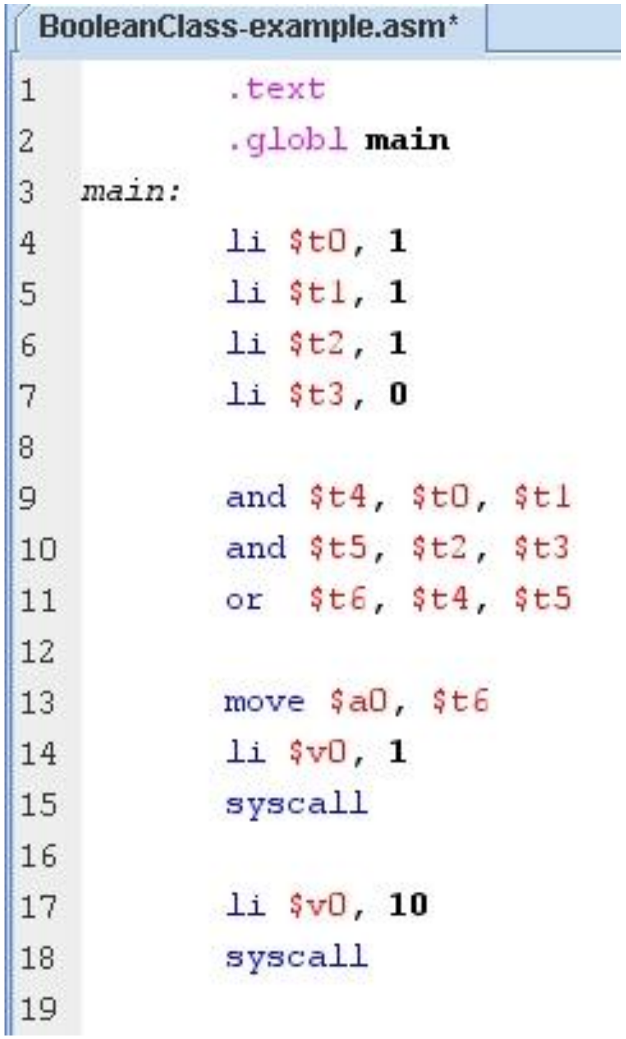
```
1
-- program is finished running --
```

In class student exercise (SOP expression)

$$X = A \bullet B + C \bullet D$$

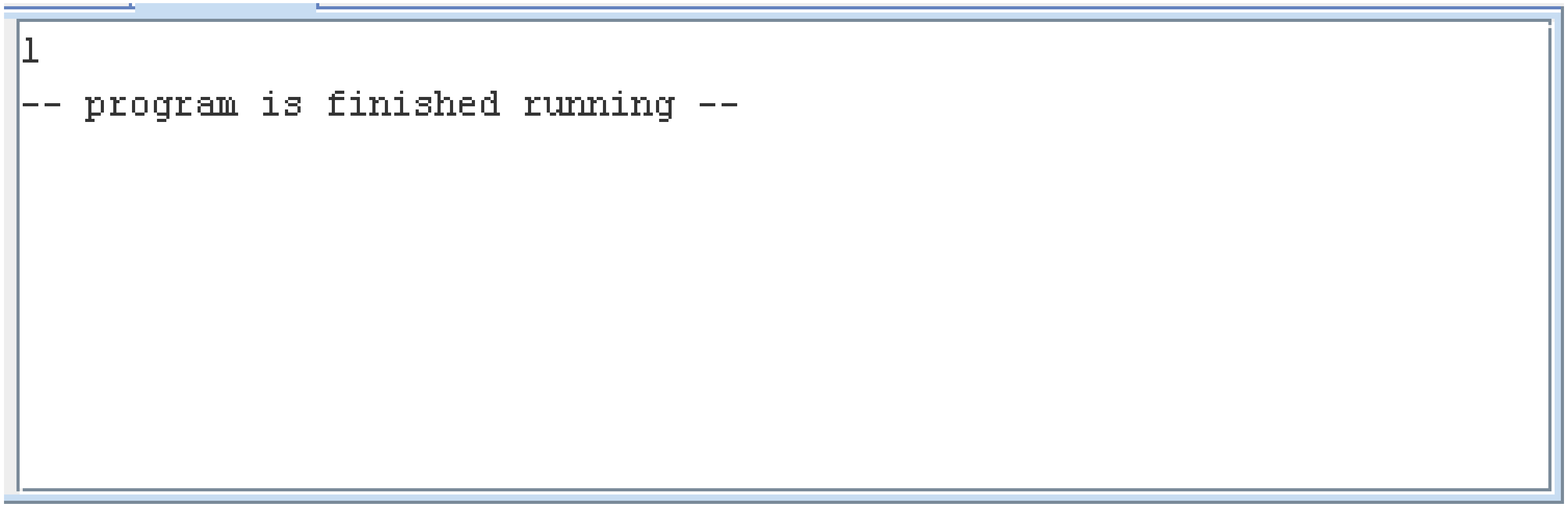
5 minutes to write the demo-code. Run it …

$X = A \bullet B + C \bullet D$



```
        .text
        .globl main
main:
        li $t0, 1
        li $t1, 1
        li $t2, 1
        li $t3, 0

        and $t4, $t0, $t1
        and $t5, $t2, $t3
        or  $t6, $t4, $t5

        move $a0, $t6
        li $v0, 1
        syscall

        li $v0, 10
        syscall

```

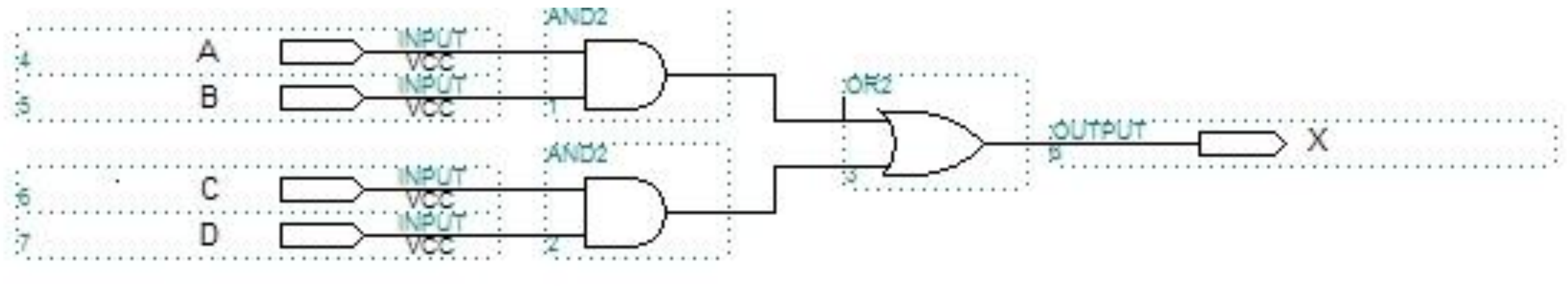
Assemble … GO



```
1
-- program is finished running --
```

Logic Circuit

$$X = A \bullet B + C \bullet D$$



AND2
4
A
INPUT
VCC
5
B
INPUT
VCC
1
OR2
OUTPUT
8
X
3
AND2
6
C
INPUT
VCC
7
D
INPUT
VCC
2