# CMP 334 (2/13/19)

Review Quiz 1 (base conversion)

Unsigned binary subtraction (example)

**H**igh **L**evel **L**anguage program compilation

**TOY** assembly language (introduction)

- **TOY** AL program execution

Representing numbers in circuits

- N < 4 circuit

Circuit → formula → truth table → circuit

Combinational circuit design process

# Quiz 1 Review

1. Convert **0x8D39** to binary.

   1000 1101 0011 1001

2. Convert **0x14A** to decimal.

   0x14A represents

$$1 \cdot 16^2 + 4 \cdot 16^1 + 10 \cdot 16^0 = 256 + 4 \cdot 16 + 10$$
$$= 256 + 64 + 10$$
$$= 330$$

# Quiz 1 Review (continued)

2. Convert **$187_{10}$** to binary.

$$
\begin{aligned}
187 \cdot 2^0 &= 93 \cdot 2^1 + 1 \cdot 2^0 \\
93 \cdot 2^1 &= 46 \cdot 2^2 + 1 \cdot 2^1 \\
46 \cdot 2^2 &= 23 \cdot 2^3 + 0 \cdot 2^2 \\
23 \cdot 2^3 &= 11 \cdot 2^4 + 1 \cdot 2^3 \\
11 \cdot 2^4 &= 5 \cdot 2^5 + 1 \cdot 2^4 \\
5 \cdot 2^5 &= 2 \cdot 2^6 + 1 \cdot 2^5 \\
2 \cdot 2^6 &= 1 \cdot 2^7 + 0 \cdot 2^6 \\
1 \cdot 2^7 &= \mathbf{0} \cdot 2^8 + 1 \cdot 2^7
\end{aligned}
$$

$$
\begin{aligned}
187_{10} &= 1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \\
&= 10111011_2
\end{aligned}
$$

# TOY Instruction Set Architecture

*Reference Card*

## Immediate values

s8 — 8-bit signed immediate
u4 — 4-bit unsigned immediate
cc — 4-bit condition code

## Register File 16 16-bit "registers"

15 real registers: $1 … $F
1 pseudo-register: $0 [$0] = 0

## Main Memory 65536 16-bit words

M[n] – n$^{th}$ memory address
^M[n] – content of M[n]

## Instructions[0]

add $T ← [$A]+[$B][1]
and $T ← [$A]&[$B][1]
bc PC ← [PC] + S8 iff CC
bcl $L ← [PC], PC ← [$A][1] iff CC
l $T ← ^M[[$A]+[$B]][1]
lwr $T ← ^M[[$A]+[$B]][1] set rsvn
lih $T$_{15..8}$ ← I8[1]
lis $T ← I8[1] (sign extended)
nor $T ← $\overline{[\$A]|[\$B]}$[1]
sl $T ← [$A]<<U4[1]
srs $T ← [$A]>>U4[1]
sru $T ← [$A]>>>U4[1]
st M[[$A]+[$B]] ← [$S]
stc M[[$A]+[$B]] ← [$S][4] iff rsvn
sub $T ← [$A]-[$B][1,2]
sys system call

### Arithmetic / Logical

| | | | | |
|---|---|---|---|---|
| 1111 | add | $T | $A | $B |
| 1110 | sub | $T | $A | $B |
| 1101 | and | $T | $A | $B |
| 1100 | nor | $T | $A | $B |

### Load /Store

| | | | | |
|---|---|---|---|---|
| 1011 | l | $T | $A | $B |
| 1010 | lwr | $T | $A | $B |
| 1001 | st | $S | $A | $B |
| 1000 | stc | $S | $A | $B |

### Shift / Branch & Link

| | | | | |
|---|---|---|---|---|
| 0111 | sru | $T | $A | U4 |
| 0110 | srs | $T | $A | U4 |
| 0101 | bcl | CC | $A | $L |
| 0100 | sl | $T | $A | U4 |

### Immediate

| | | | |
|---|---|---|---|
| 0011 | lih | $T | S8 |
| 0010 | lis | $T | S8 |
| 0001 | bc | CC | S8 |
| 0000 | sys | $X | S8 |

### Condition Codes

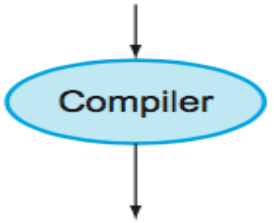
| | | |
|---|---|---|
| 1111 | $\overline{znv}$ + nv | SGT |
| 1110 | v | OVF |
| 1101 | n$\bar{v}$ + $\bar{n}$v | SLT |
| 1100 | n | NEG |
| 1011 | $\bar{z}$c | UGT |
| 1010 | $\bar{z}$ | NE |
| 1001 | $\bar{c}$ | ULT |
| 1000 | 0 | NOP |
| 0111 | z + n$\bar{v}$ + $\bar{n}$v | SLE |
| 0110 | $\bar{v}$ | NVF |
| 0101 | nv + $\overline{nv}$ | SGE |
| 0100 | $\bar{n}$ | POS |
| 0011 | z + $\bar{c}$ | ULE |
| 0010 | z | EQ |
| 0001 | c | UGE |
| 0000 | 1 | ALL |

## Notes

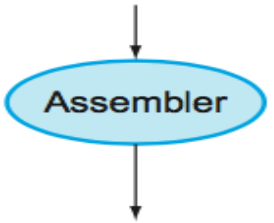
[0] PC ← PC+1 ***before*** instruction execution
[1] $0 ***not*** changed
[2] Determines flags: z, n, c, v
[4] Determines flag: v

# Assignment Compilation

$$Z = X + Y$$

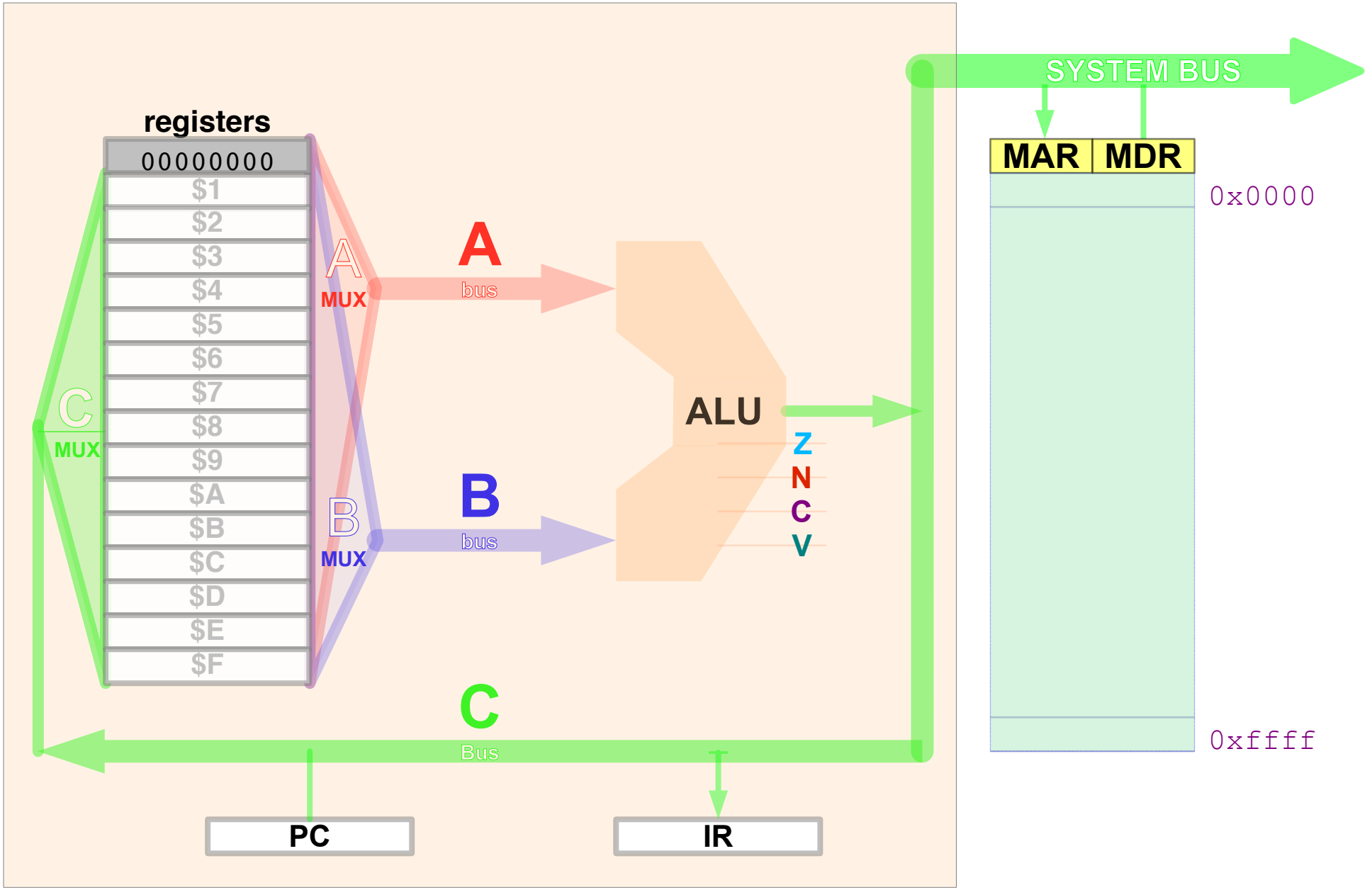



```
l   $5, $A, $0     : $5   = M[X]
l   $6, $B, $0     : $6   = M[Y]
add $7, $5, $6     : $7   = M[X] + M[Y]
st  $7, $C, $0     : M[Z] = M[X] + M[Y]
```



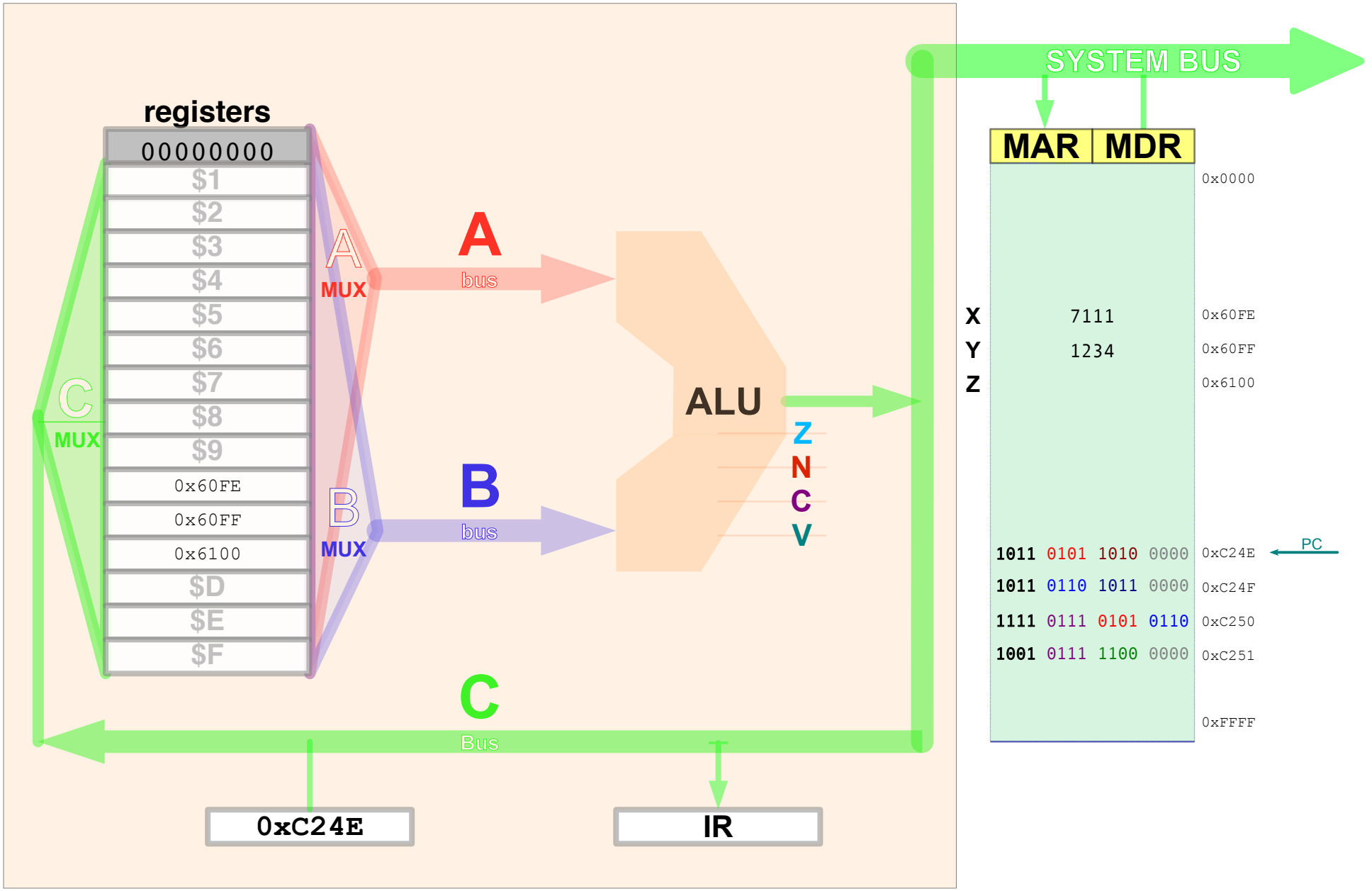

```
1011 0101 1010 0000 1011 0110 1011 0000
1111 0111 0101 0110 1001 0111 1100 0000
```

# TOY Processor & RAM



# The TOY Computer

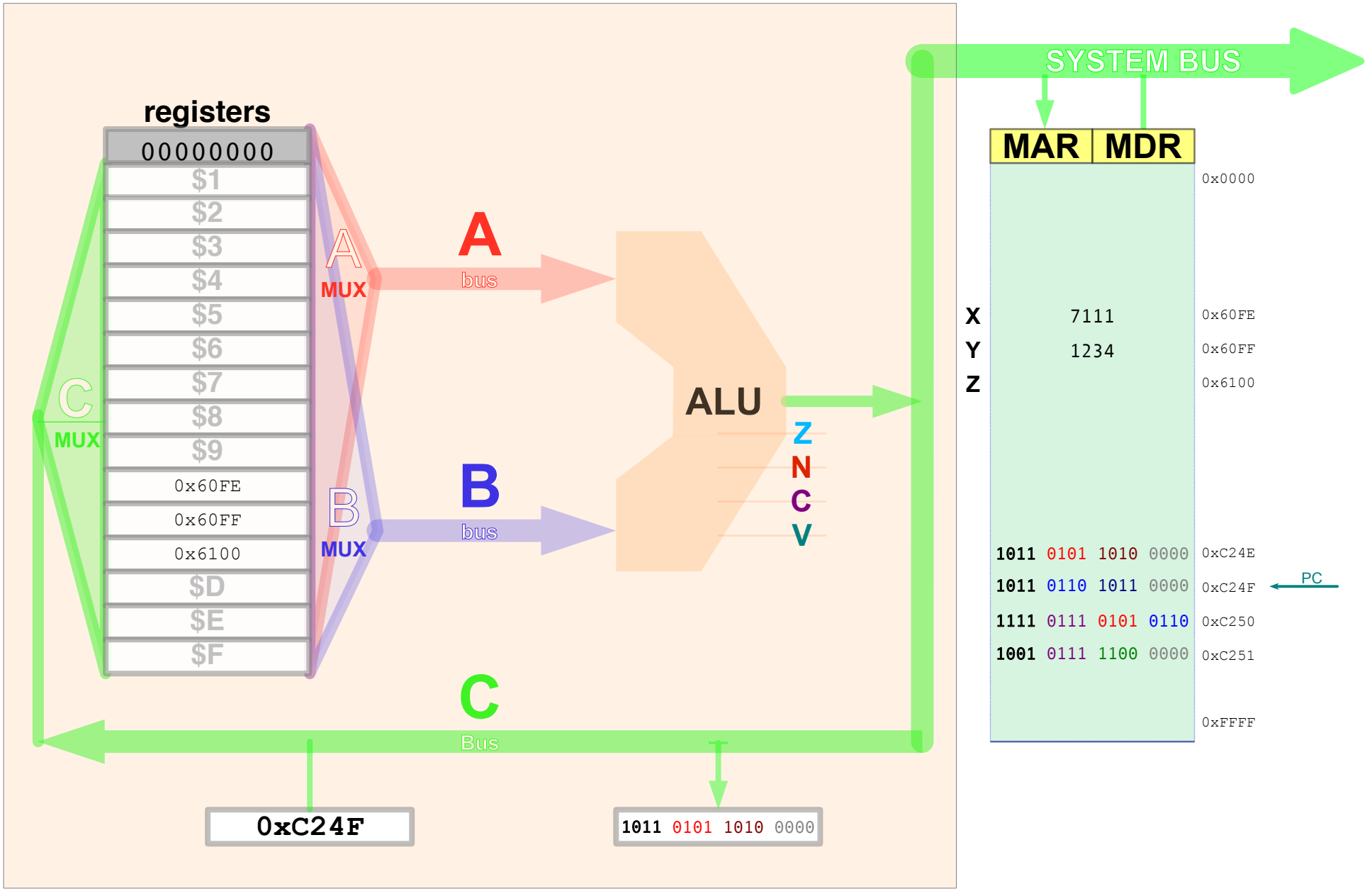
fetch



fetch

# The TOY Computer

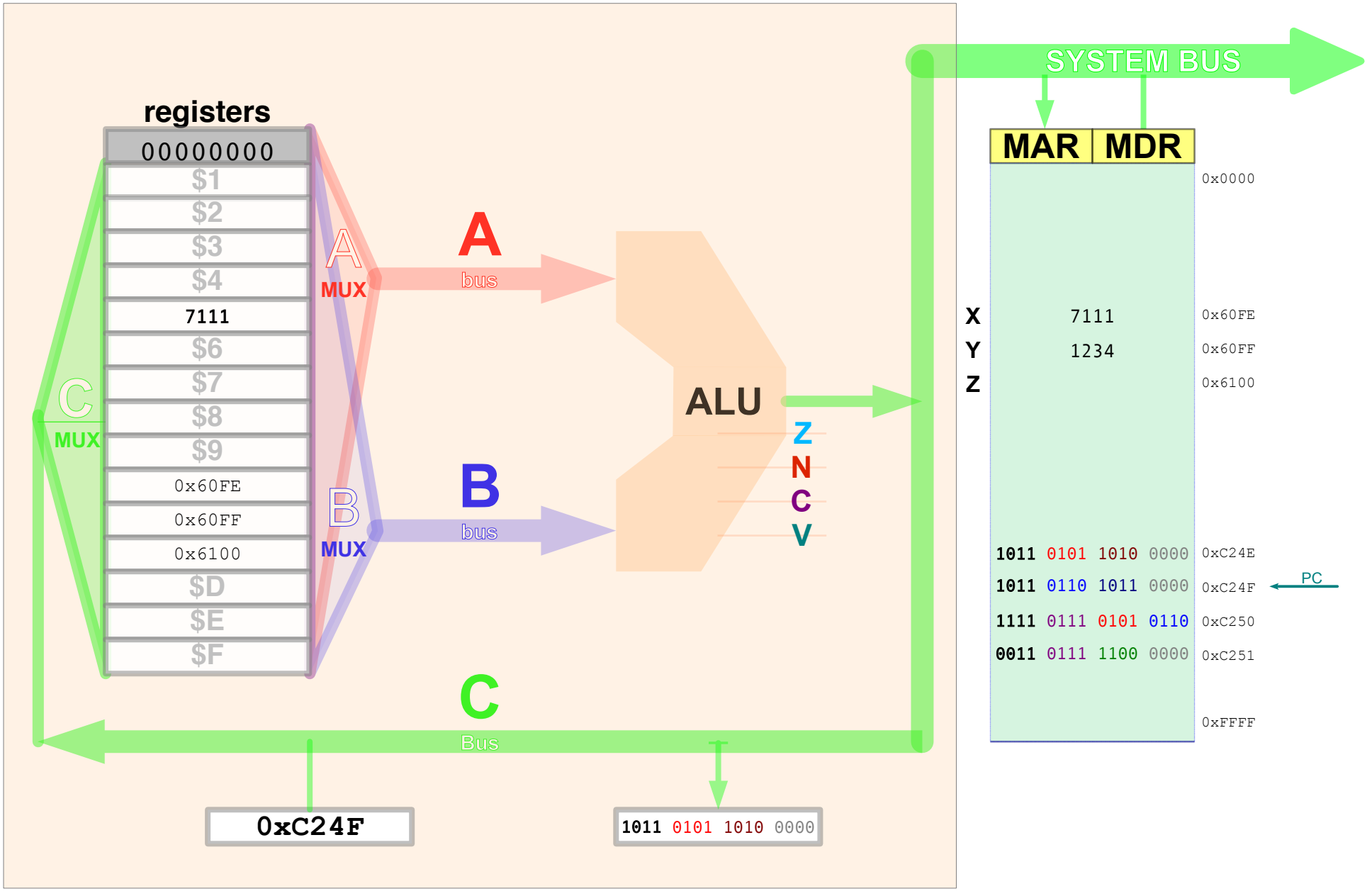
execute



execute

# The TOY Computer

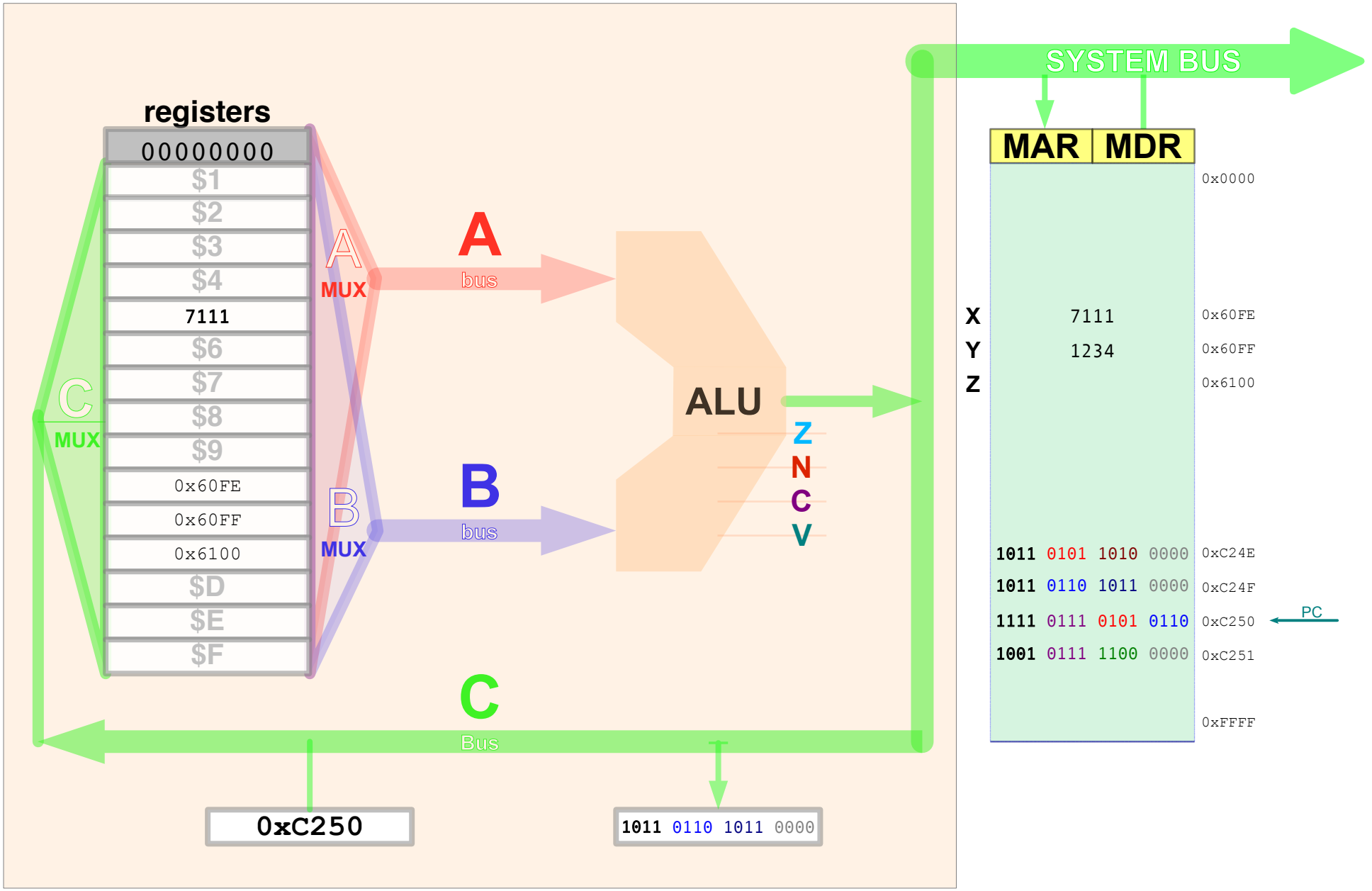
fetch



fetch

# The TOY Computer

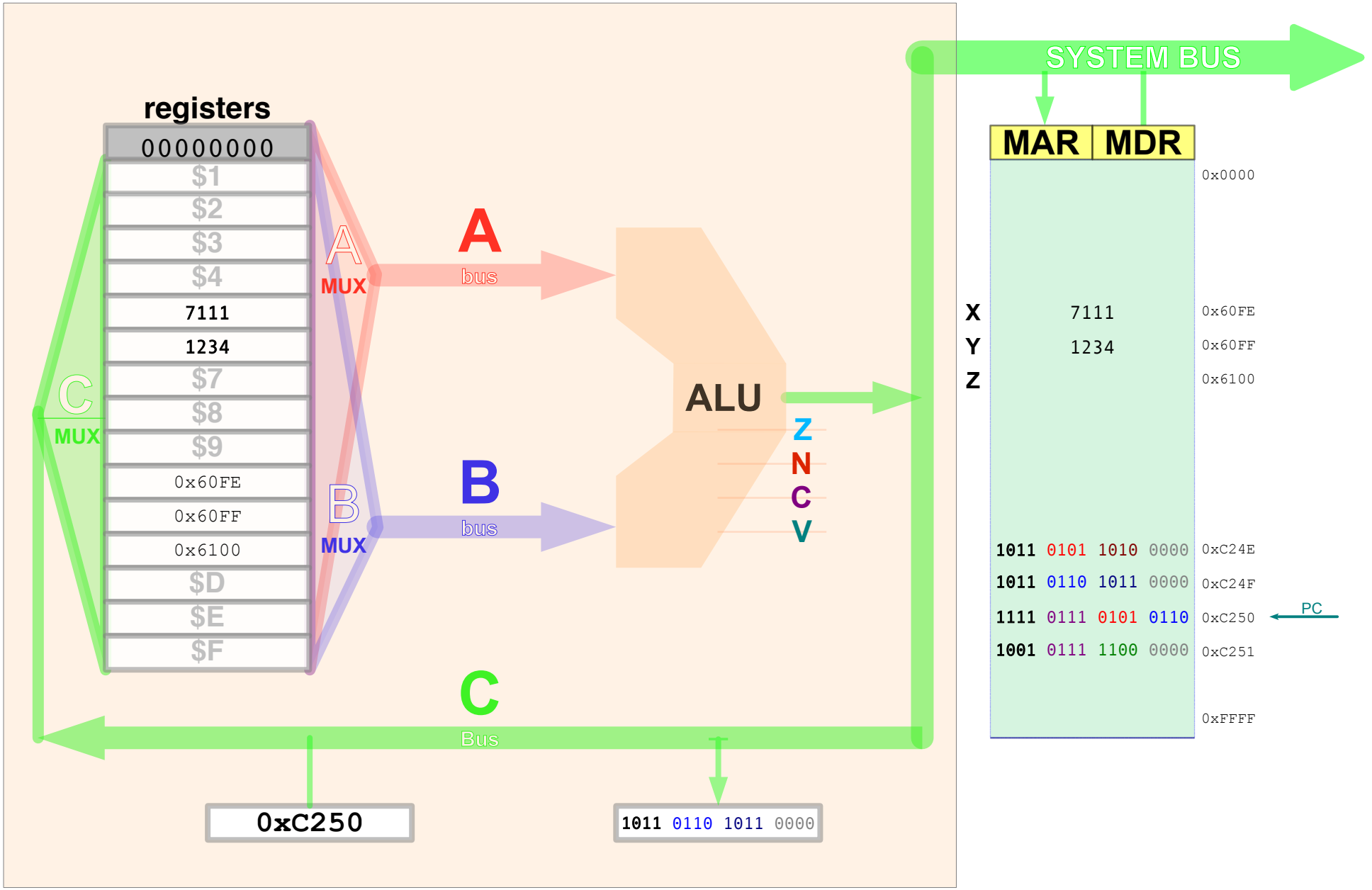
execute



execute

# The TOY Computer

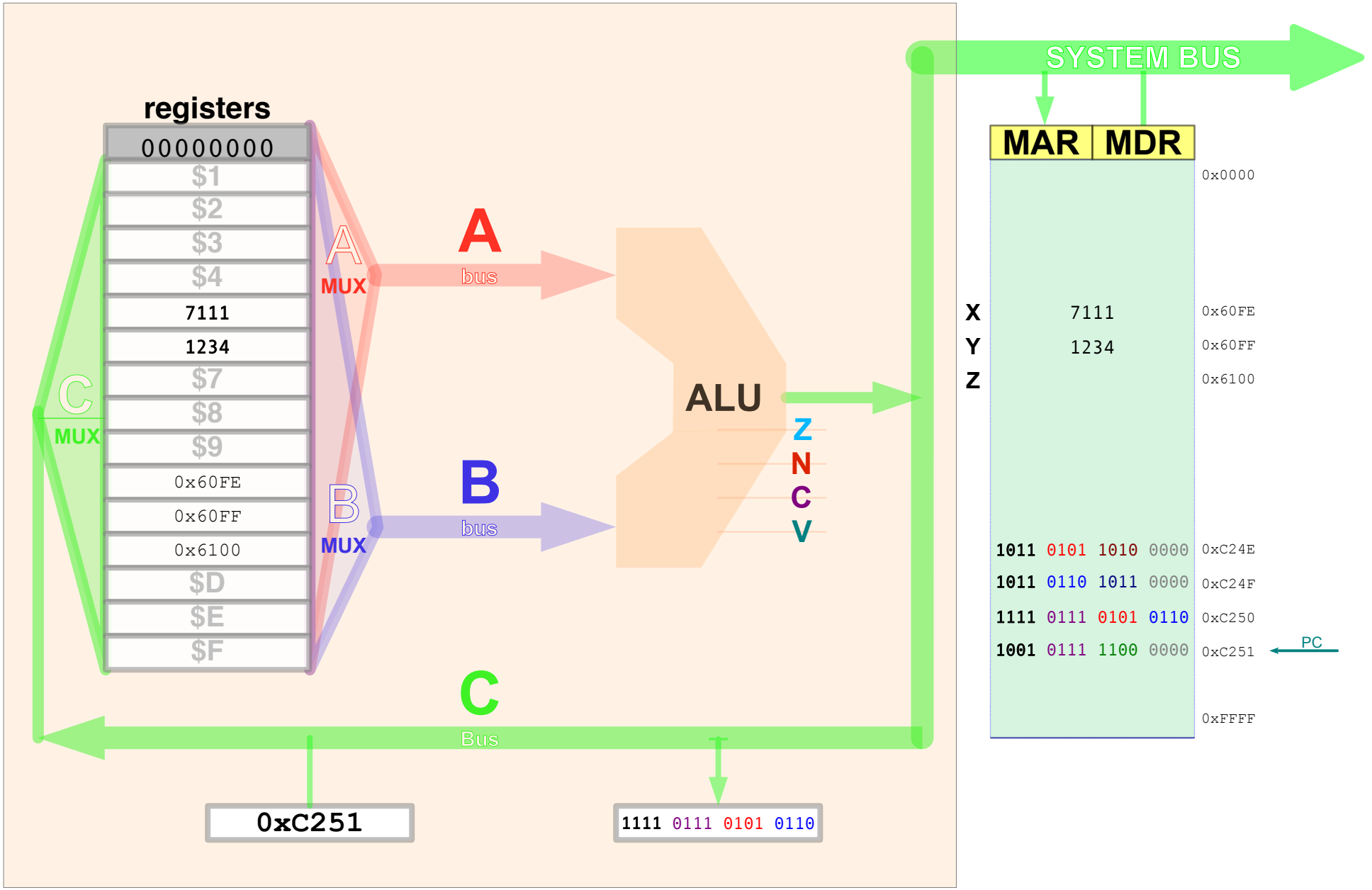
fetch



fetch

# The TOY Computer

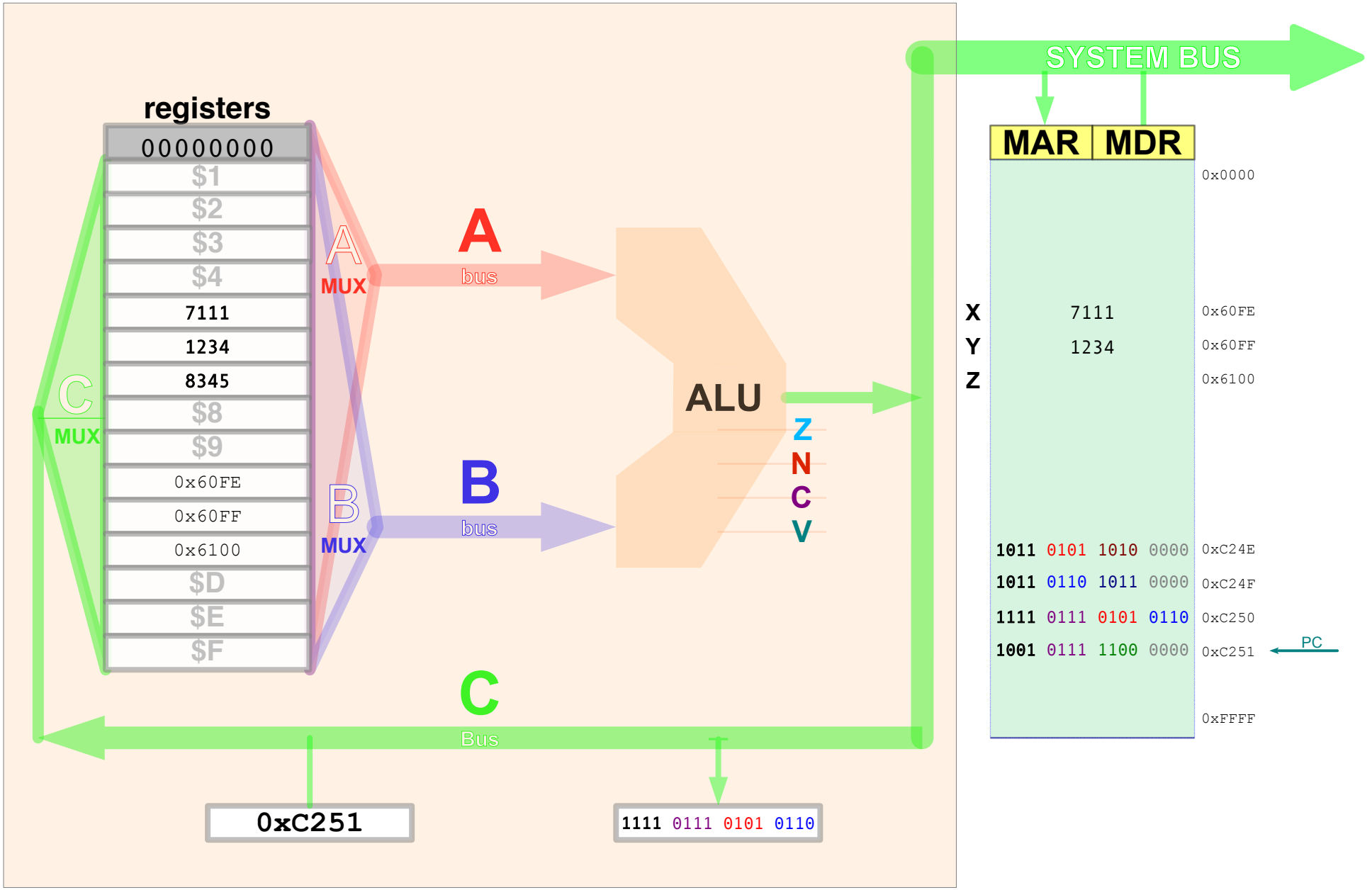
execute



execute

# The TOY Computer

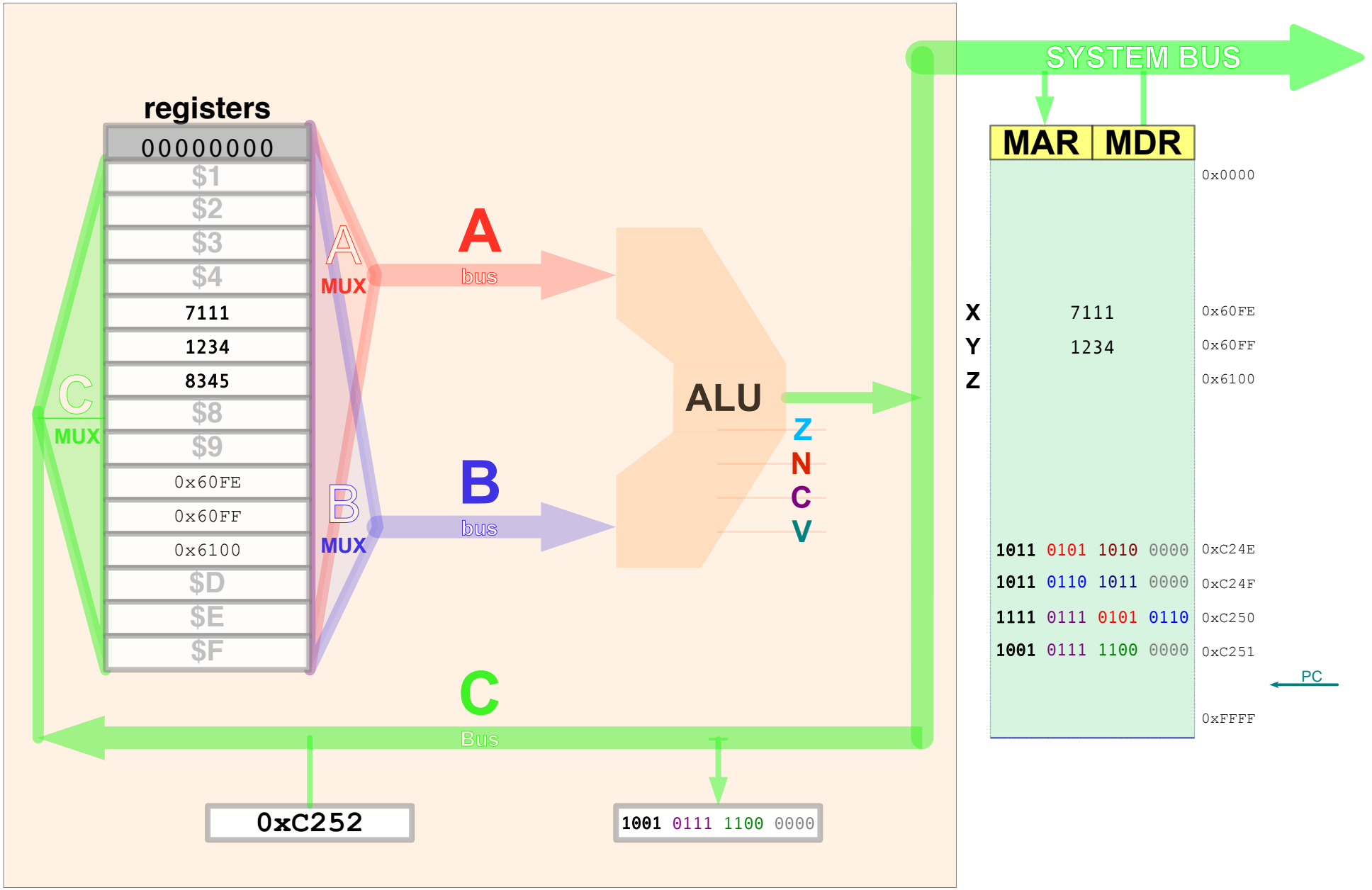
fetch



fetch

# The TOY Computer

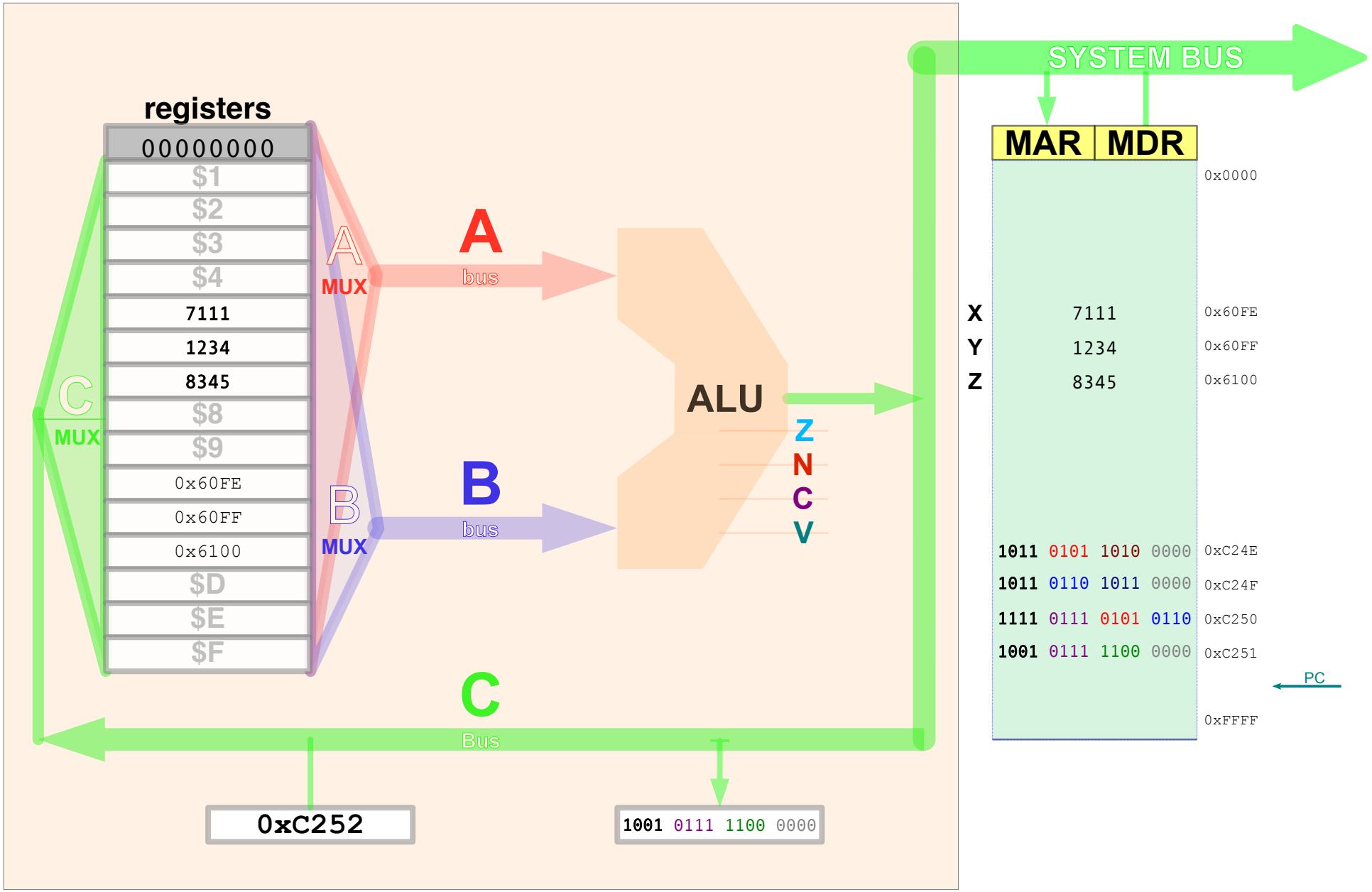
execute



execute

# The TOY Computer

fetch



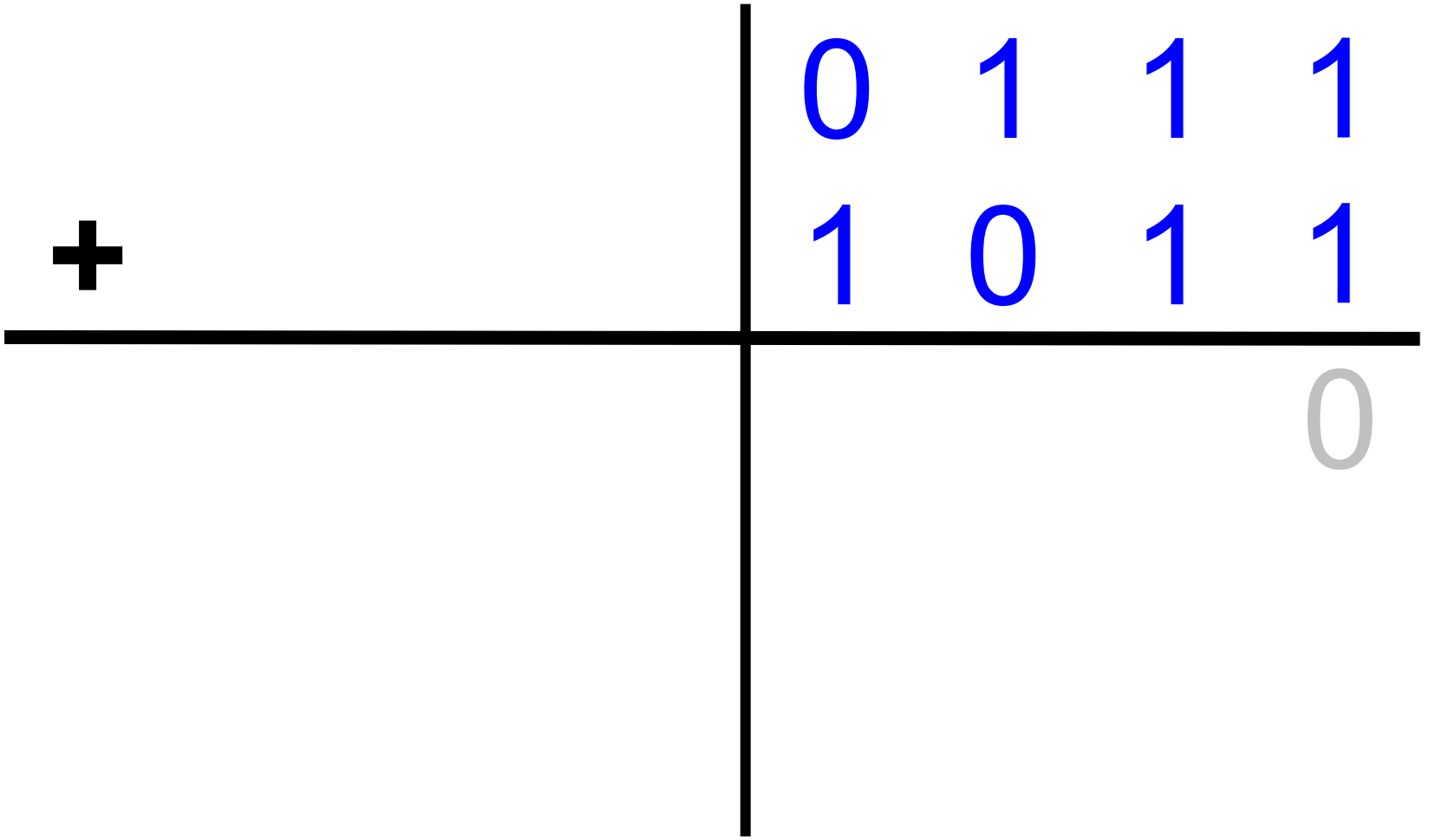
# HW 7: `W` = `X` + `Y` + `Z`

Assignment: Write a **TOY** assembly language program to add the values of 3 variables, **`X`**, **`Y`**, and **`Z`**, in memory and store the sum in a fourth, **`W`**.

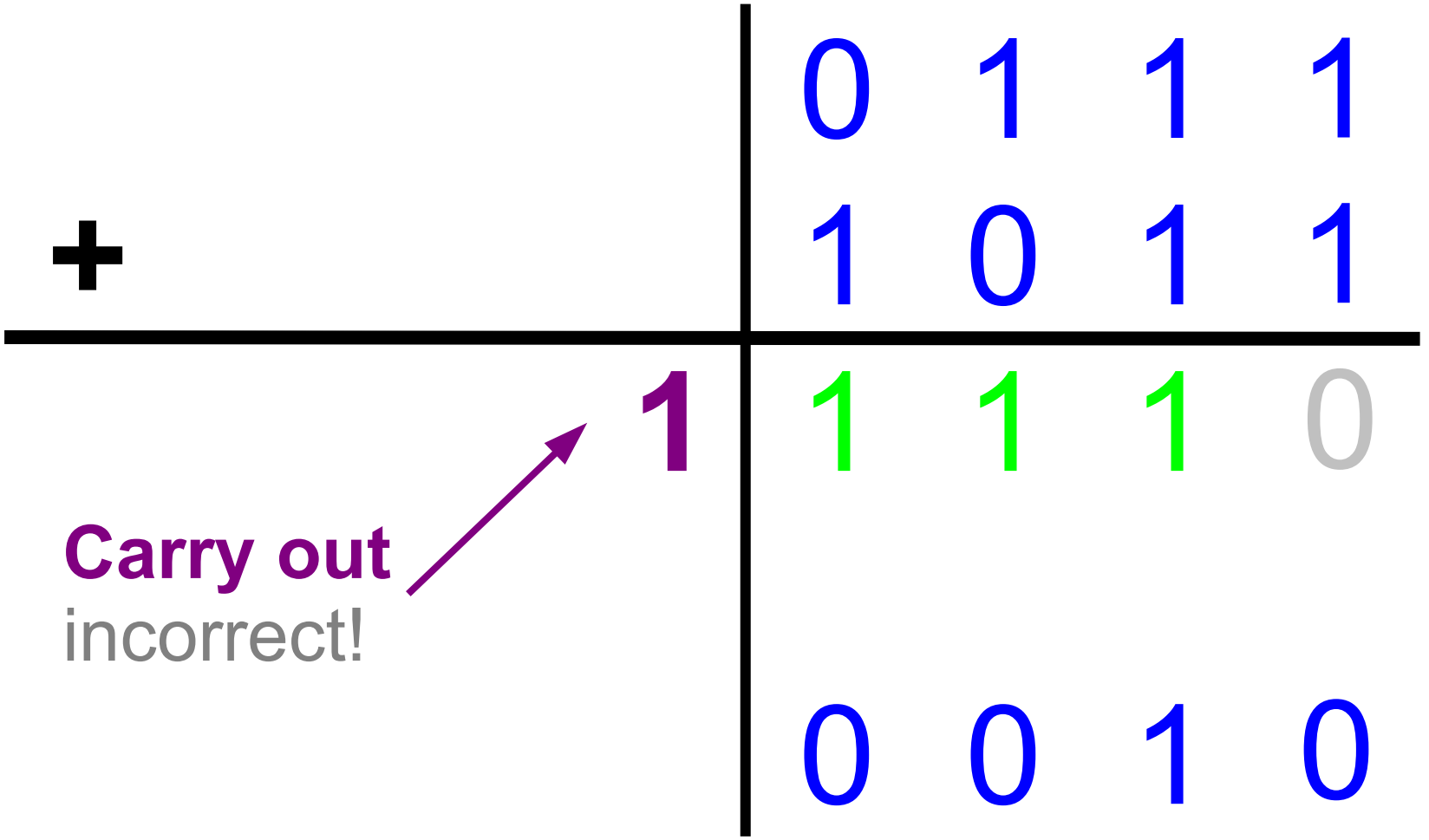
Details: The variables occupy consecutive words of memory starting with **`W`**. The address of **`W`** is in register **`$3`**. Register **`$4`** contains the constant **`1`**. *Do not change* the values in registers **`$0`** through **`$4`**. You can use registers **`$5`** through **`$F`** as you please. Your program should consist entirely of addition (**`add`**), load (**`l`**), and store (**`st`**) instructions.

Hint: You will need to construct the addresses of variables **`X`**, **`Y`**, and **`Z`** in a register or in registers. There are various ways to accomplish this.

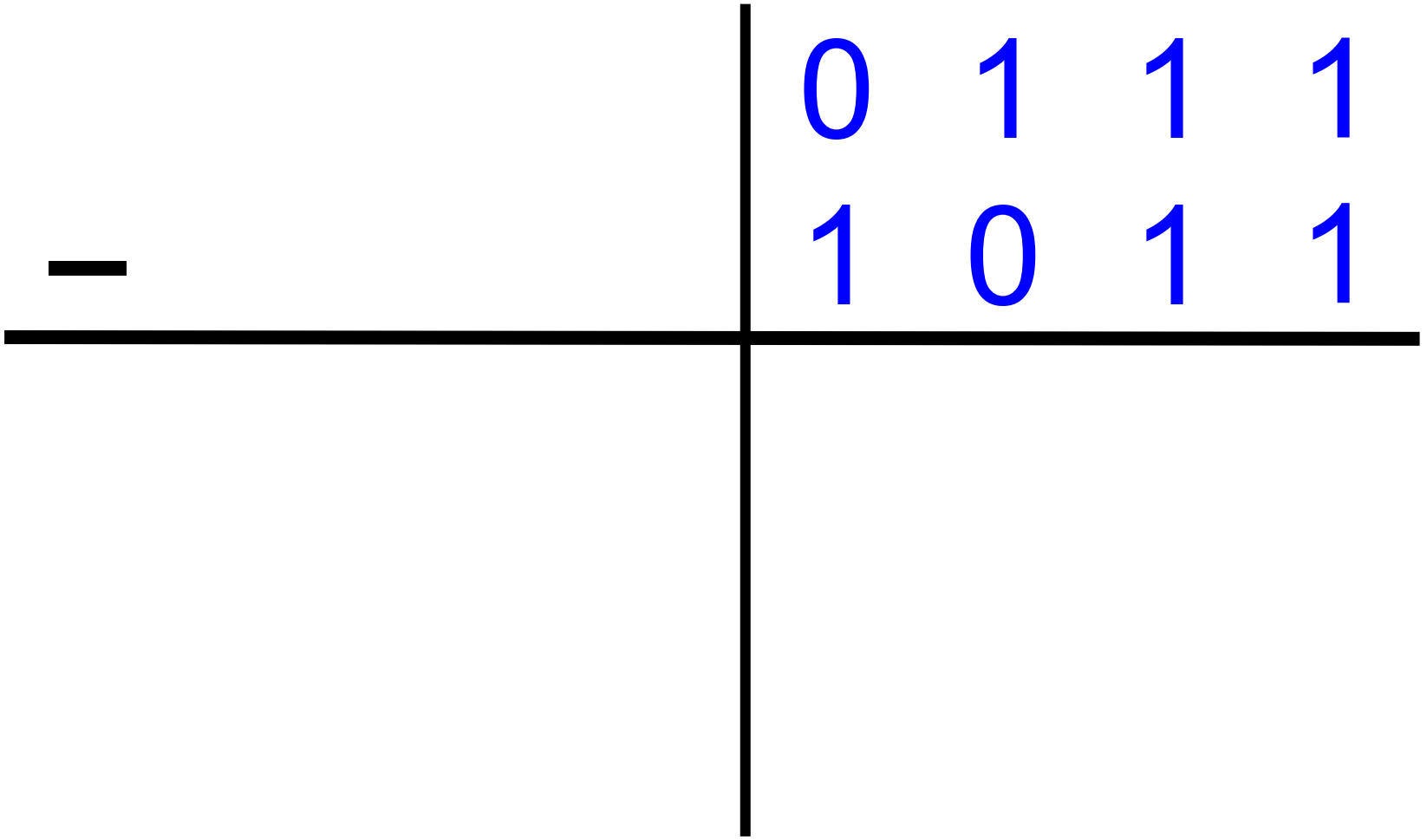
# 7 + B = ?



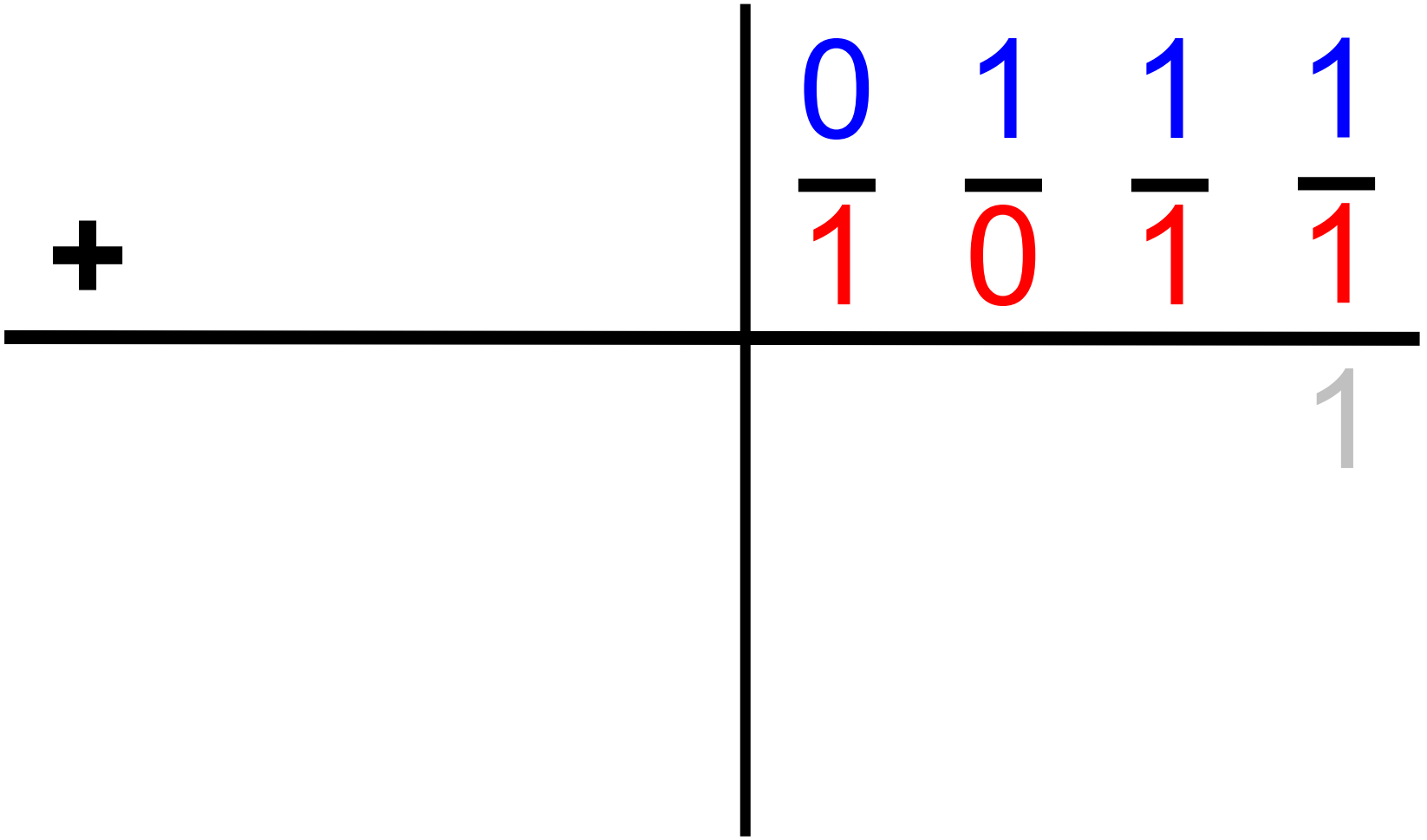
# 7 + B ≠ 2



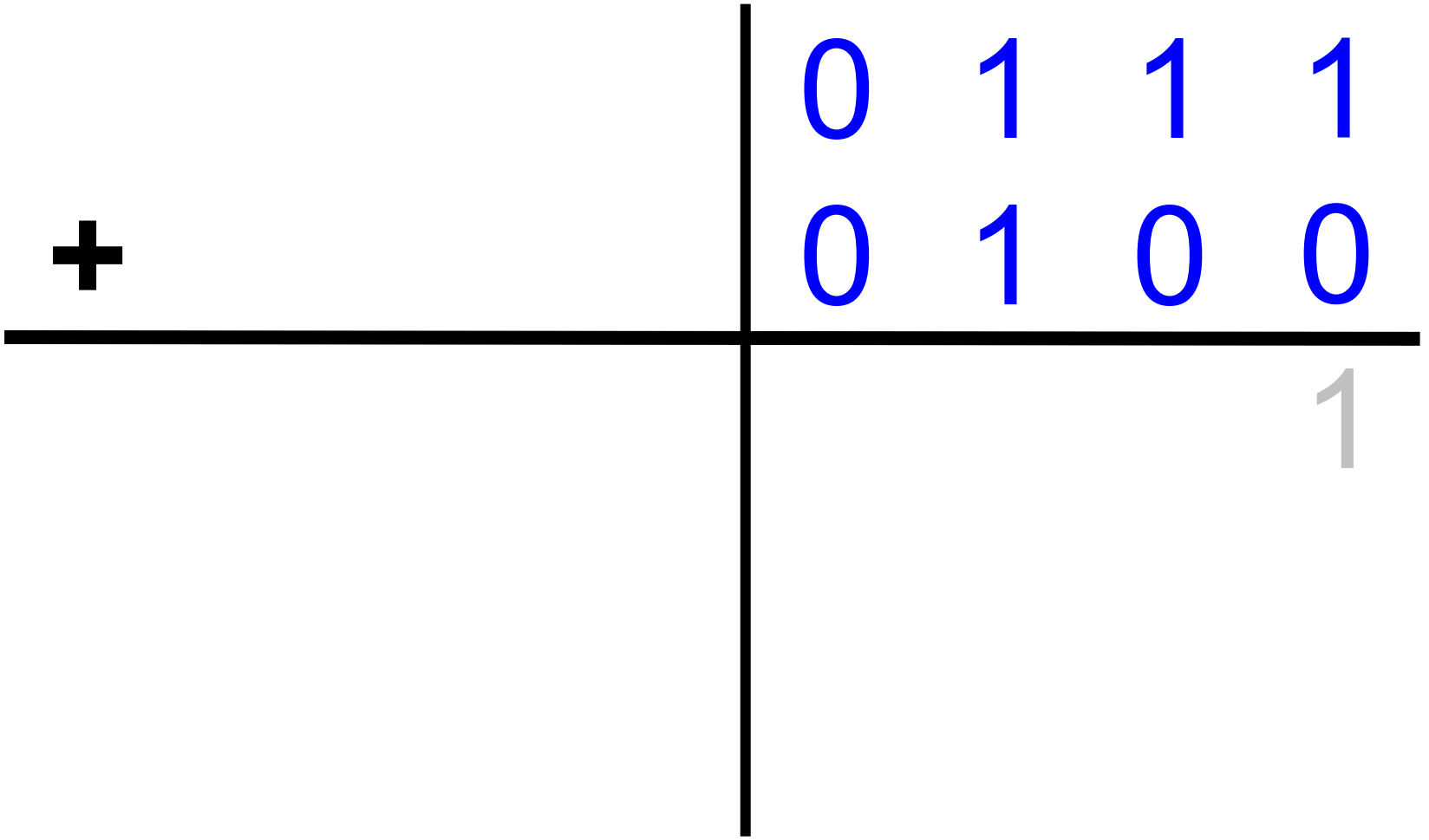
# 7 – B = ?



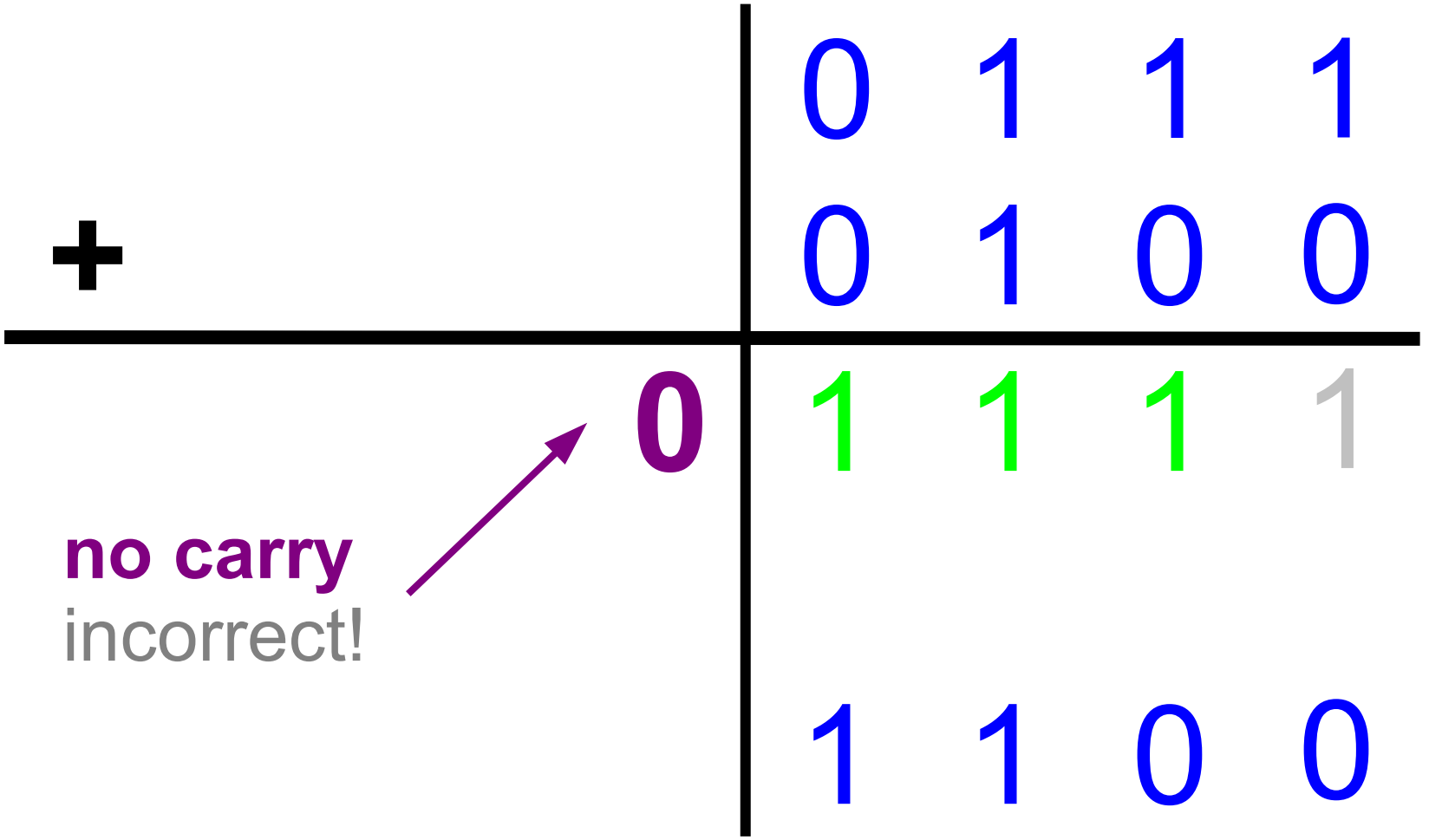
# 7 – B = ?



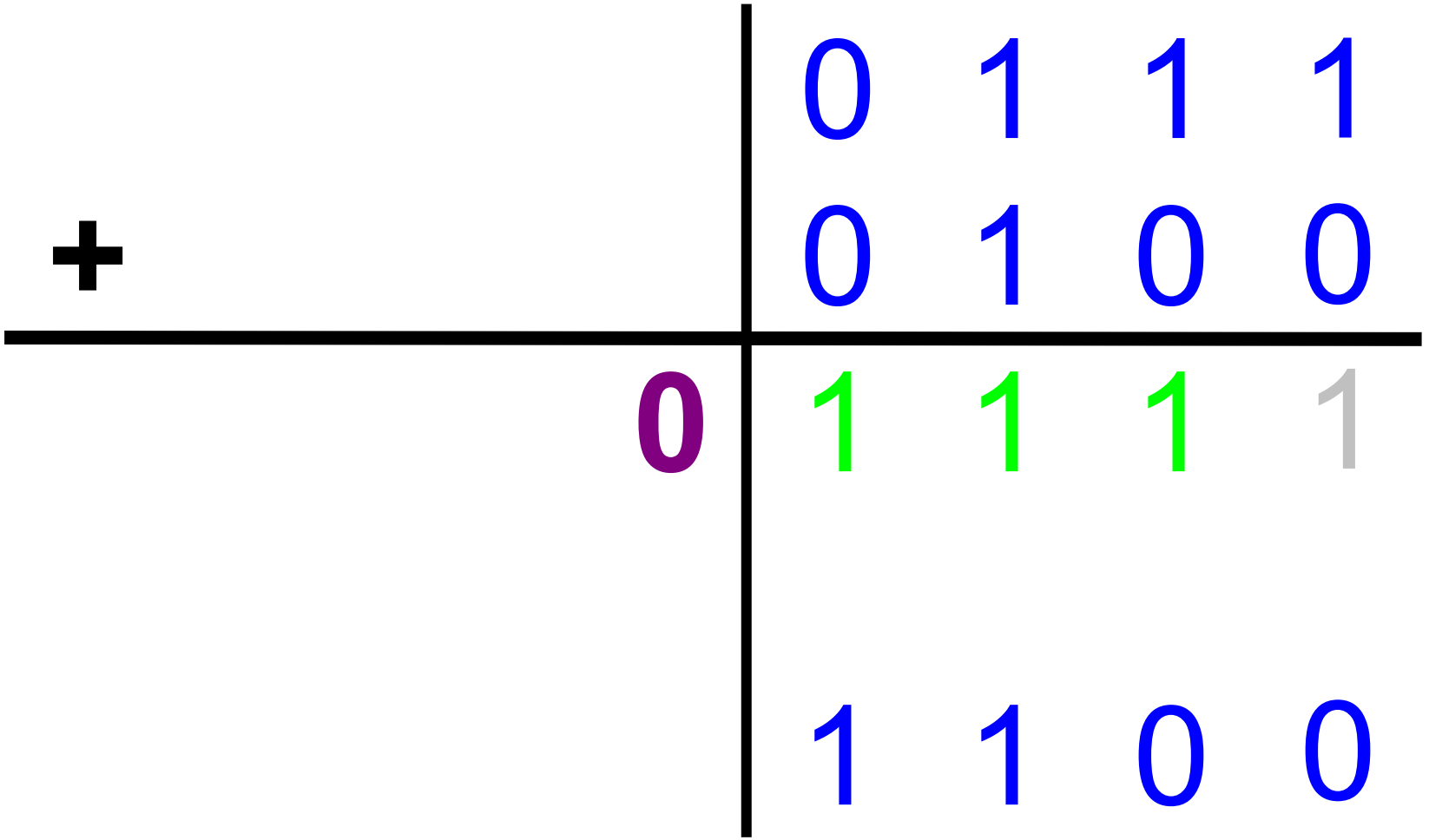
# 7 – B = ?



# 7 – B ≠ C



# $7 - B = C - 2^4 = -4$



# A – C + 3 = A + 3 – C

| A – C + 3 | A + 3 – C |
| --- | --- |
| = 1010 – 1100 + 0011 | = 1010 + 0011 – 1100 |
| = 1010 + 0100 + 0011 | = 1101 – 1100 |
| = 1110 + 0011 | = 1101 + 0100 |
| = 0001 | = 0001 |

# Circuit Design Example: $n > 4$

Construct a circuit with Boolean inputs $\mathbf{n}_2$, $\mathbf{n}_1$, and $\mathbf{n}_0$ that is 1 if, and only if, $n \equiv \mathbf{n}_2 \cdot 2^2 + \mathbf{n}_1 \cdot 2^1 + \mathbf{n}_0 \cdot 2^0 > 4$.

| $\mathbf{n}_2$ | $\mathbf{n}_1$ | $\mathbf{n}_0$ | $\mathbf{n}$ | $\mathbf{n} > 4$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 2 | 0 |
| 0 | 1 | 1 | 3 | 0 |
| 1 | 0 | 0 | 4 | 0 |
| 1 | 0 | 1 | 5 | 1 |
| 1 | 1 | 0 | 6 | 1 |
| 1 | 1 | 1 | 7 | 1 |

# Circuit Design Example: $n > 4$

Construct a circuit with Boolean inputs $\mathbf{n}_2$, $\mathbf{n}_1$, and $\mathbf{n}_0$ that is 1 if, and only if, $n \equiv \mathbf{n}_2 \cdot 2^2 + \mathbf{n}_1 \cdot 2^1 + \mathbf{n}_0 \cdot 2^0 > 4$.

| $\mathbf{n}_2$ | $\mathbf{n}_1$ | $\mathbf{n}_0$ | $\mathbf{n}$ | $\mathbf{n} > 4$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 2 | 0 |
| 0 | 1 | 1 | 3 | 0 |
| 1 | 0 | 0 | 4 | 0 |
| 1 | 0 | 1 | 5 | 1 |
| 1 | 1 | 0 | 6 | 1 |
| 1 | 1 | 1 | 7 | 1 |

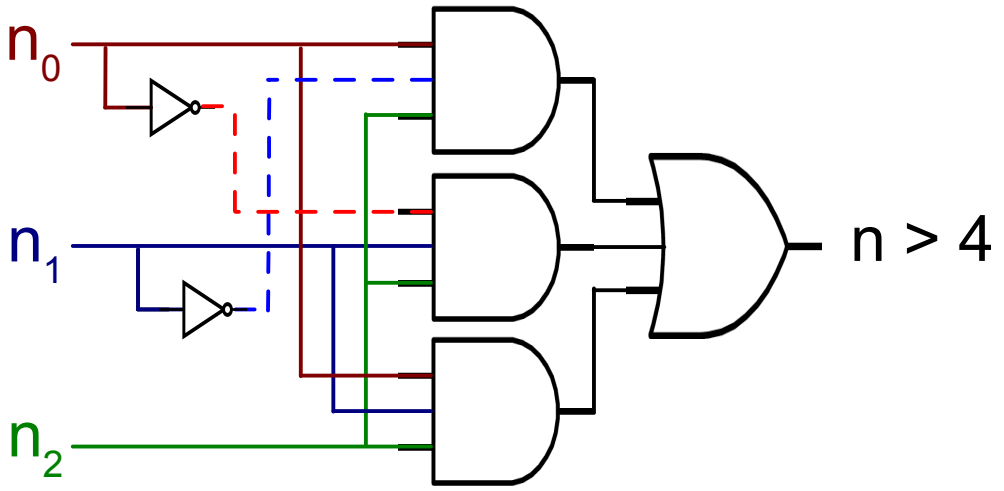
$$n_2\bar{n}_1n_0 + n_2n_1\bar{n}_0 + n_2n_1n_0$$

# Circuit Design Example: $n > 4$

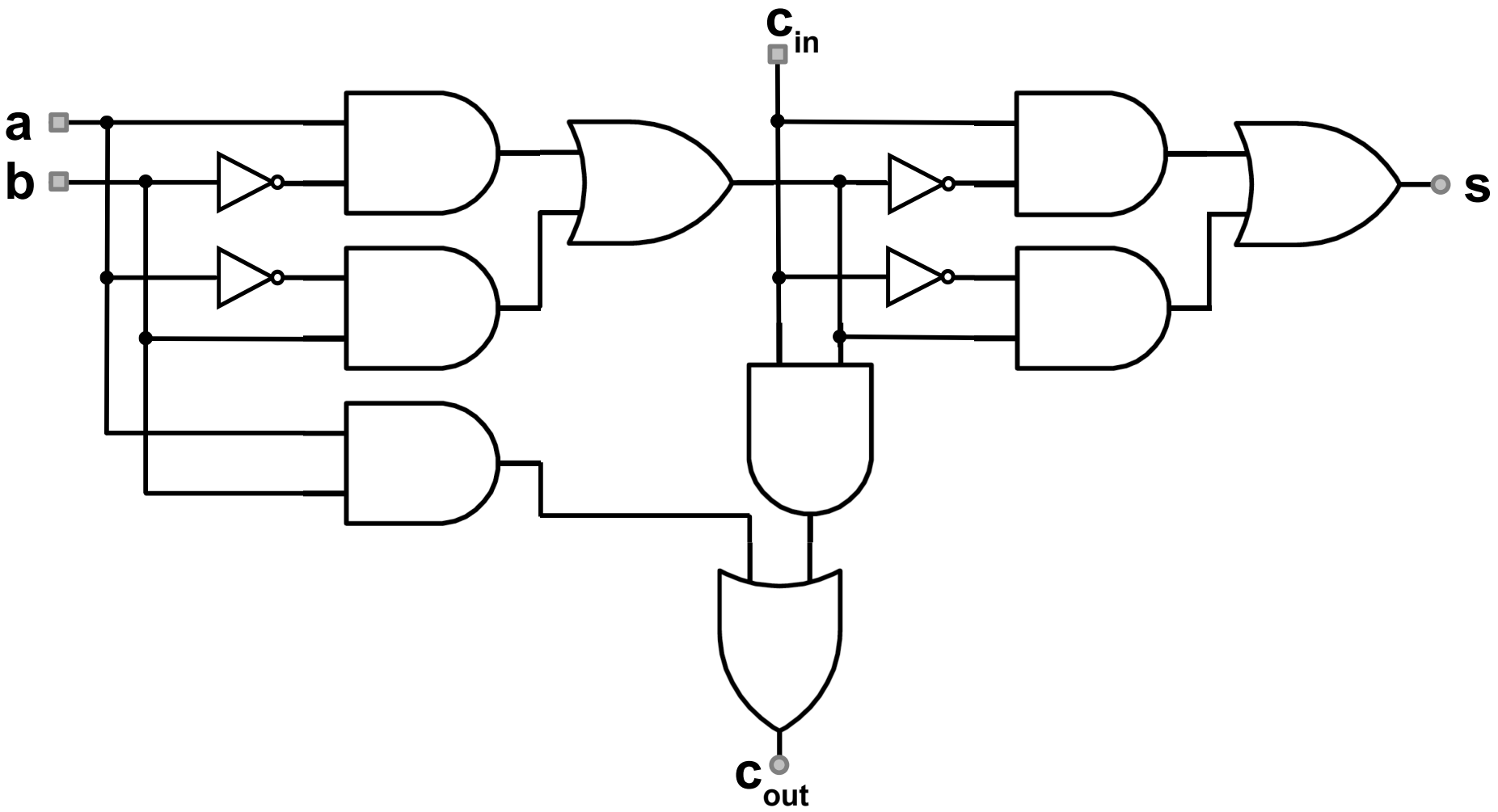
Construct a circuit with Boolean inputs $\mathbf{n}_2$, $\mathbf{n}_1$, and $\mathbf{n}_0$ that is 1 if, and only if, $n \equiv \mathbf{n}_2 \cdot 2^2 + \mathbf{n}_1 \cdot 2^1 + \mathbf{n}_0 \cdot 2^0 > 4$.

| $\mathbf{n}_2$ | $\mathbf{n}_1$ | $\mathbf{n}_0$ | $\mathbf{n}$ | $\mathbf{n} > 4$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 2 | 0 |
| 0 | 1 | 1 | 3 | 0 |
| 1 | 0 | 0 | 4 | 0 |
| 1 | 0 | 1 | 5 | 1 |
| 1 | 1 | 0 | 6 | 1 |
| 1 | 1 | 1 | 7 | 1 |

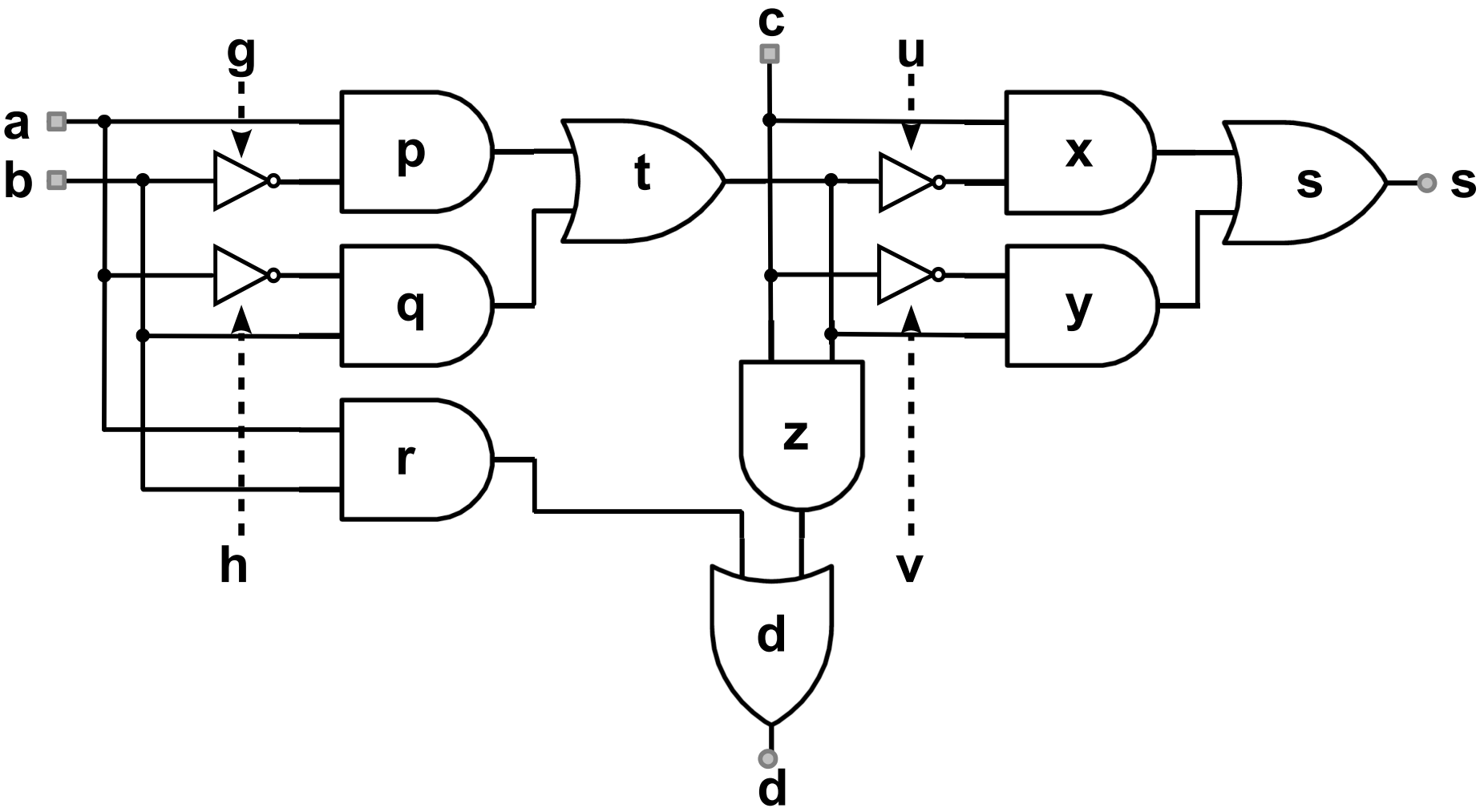
$$n_2\bar{n}_1n_0 + n_2n_1\bar{n}_0 + n_2n_1n_0$$



# Combinational Circuit → Boolean Formula

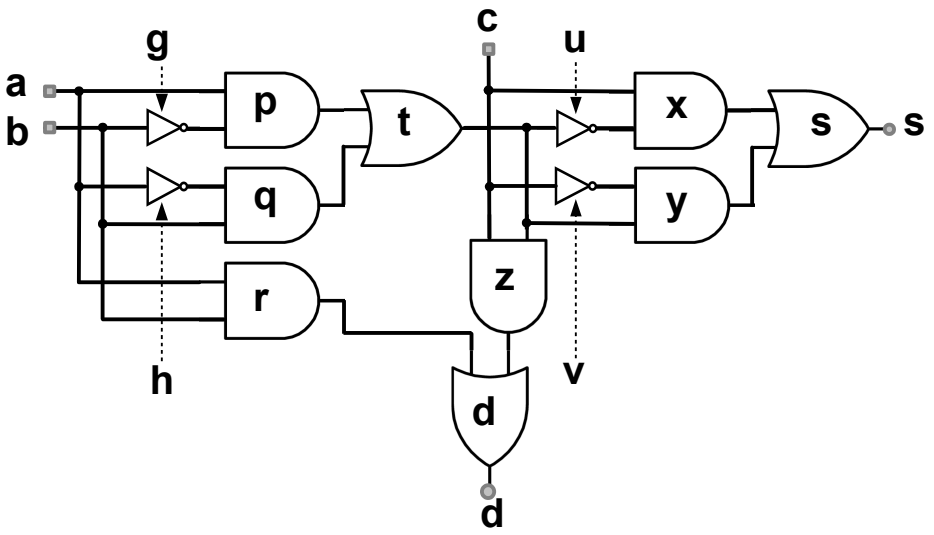


# Assign a Variable for each Gate



# Circuit → Formula

```
s = x + y
x = uc
y = tv
```



# Circuit → Formula

$s = x + y$

$x = uc$

$y = t\bar{v}$

$u = \bar{t}$

$v = \bar{c}$

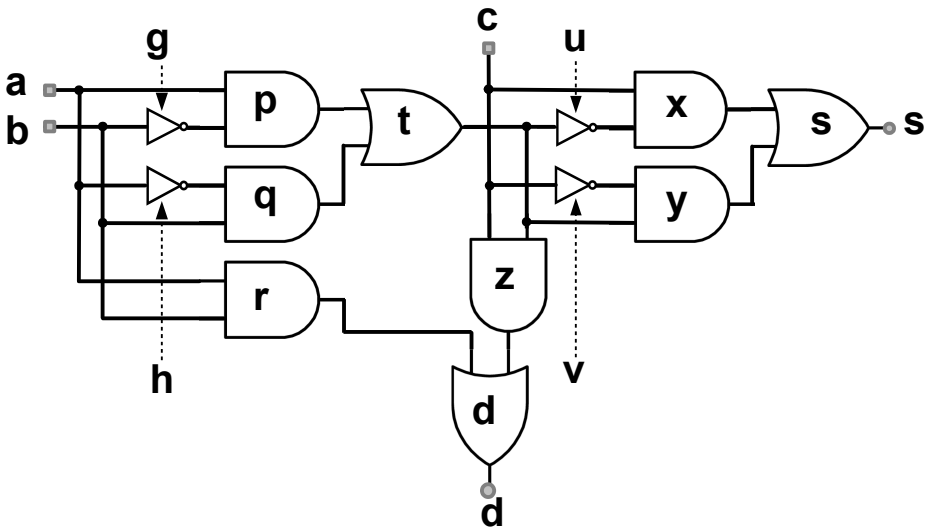
$t = p + q$

$p = ag$

$q = h\bar{b}$

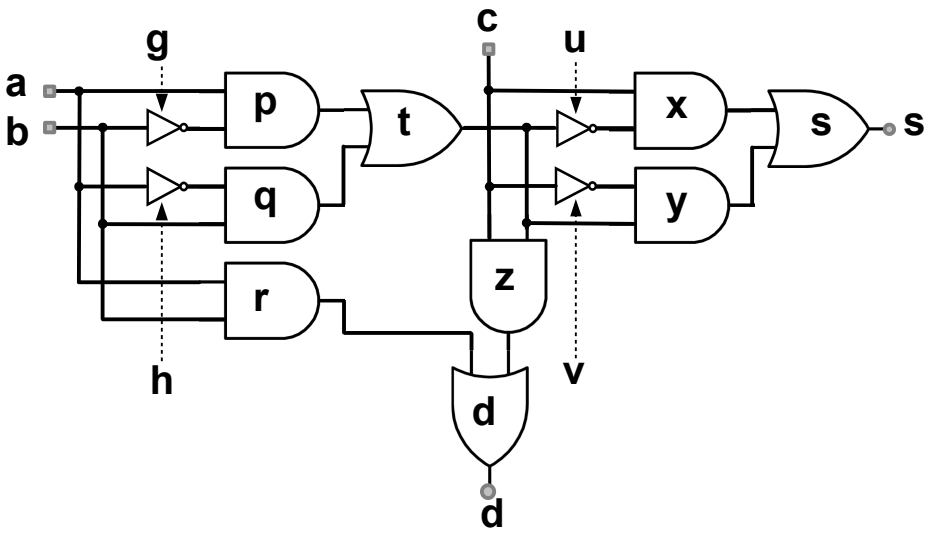
$g = \bar{b}$

$h = \bar{a}$



# Circuit → Formula

$$
\begin{aligned}
s &= x + y \\
x &= uc \\
y &= tv \\
u &= \overline{t} \\
v &= \overline{c} \\
t &= p + q \\
p &= ag \\
q &= h\overline{b} \\
g &= \overline{b} \\
h &= \overline{a} \\
t &= a\overline{b} + \overline{a}b \\
s &= t\overline{c} + \overline{t}c \\
&= a\overline{bc} + \overline{a}b\overline{c} + \overline{t}c \\
&= a\overline{bc} + \overline{a}b\overline{c} + \\
&\quad \sim(a\overline{b} + \overline{a}b)\cdot c
\end{aligned}
$$



# Boolean Formula → Circuit

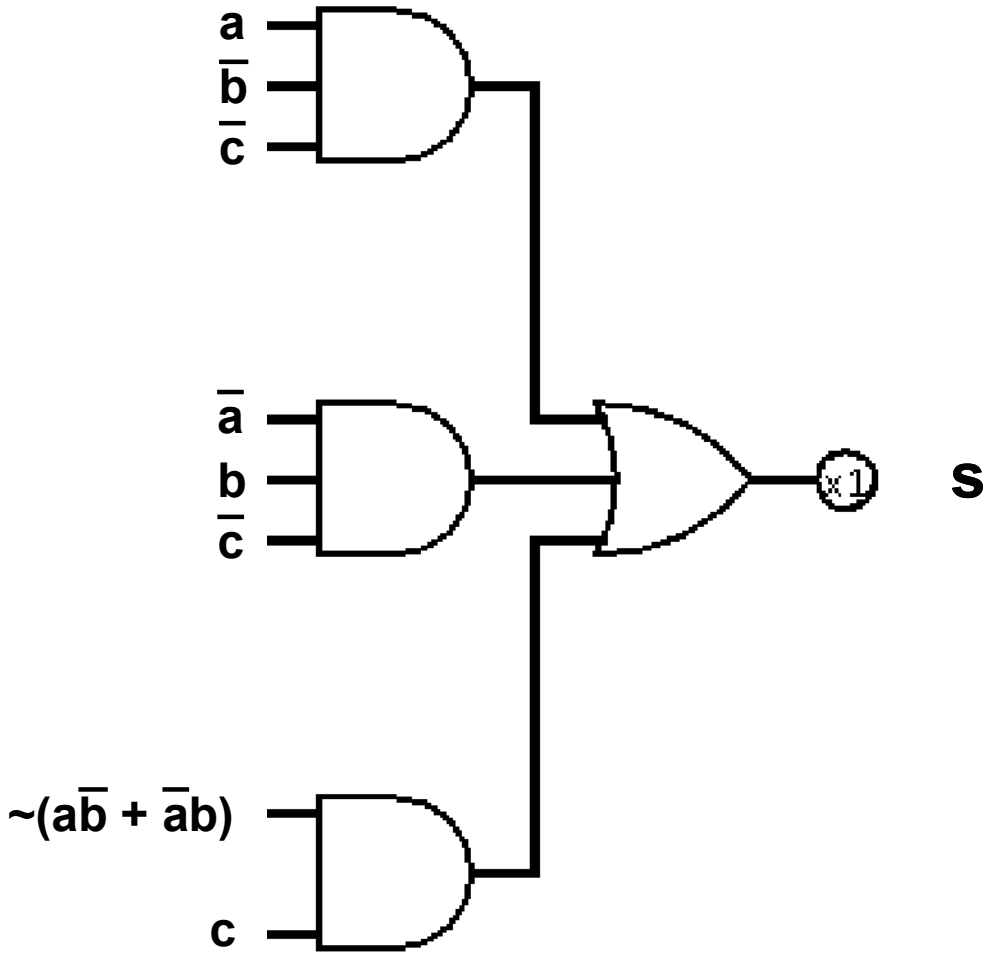
$$s = a\overline{bc} + \bar{a}b\bar{c} + \sim(a\bar{b} + \bar{a}b)\cdot c$$

# Boolean Formula → Circuit

$$s = a\overline{bc} + \bar{a}b\bar{c} + \sim(a\bar{b} + \bar{a}b)\cdot c$$

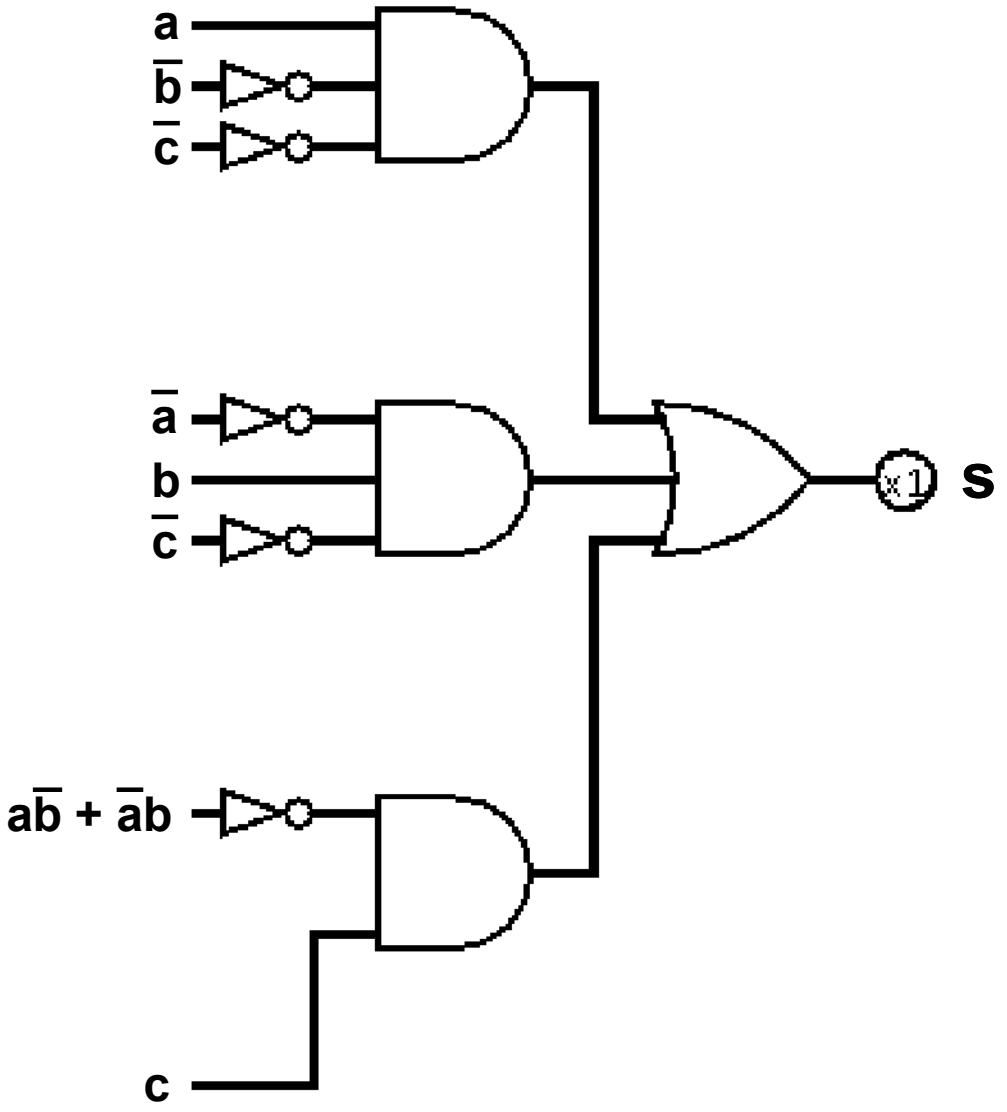
# Boolean Formula → Circuit

$$s = a\bar{b}\bar{c} + \bar{a}b\bar{c} + \sim(a\bar{b} + \bar{a}b)\cdot c$$



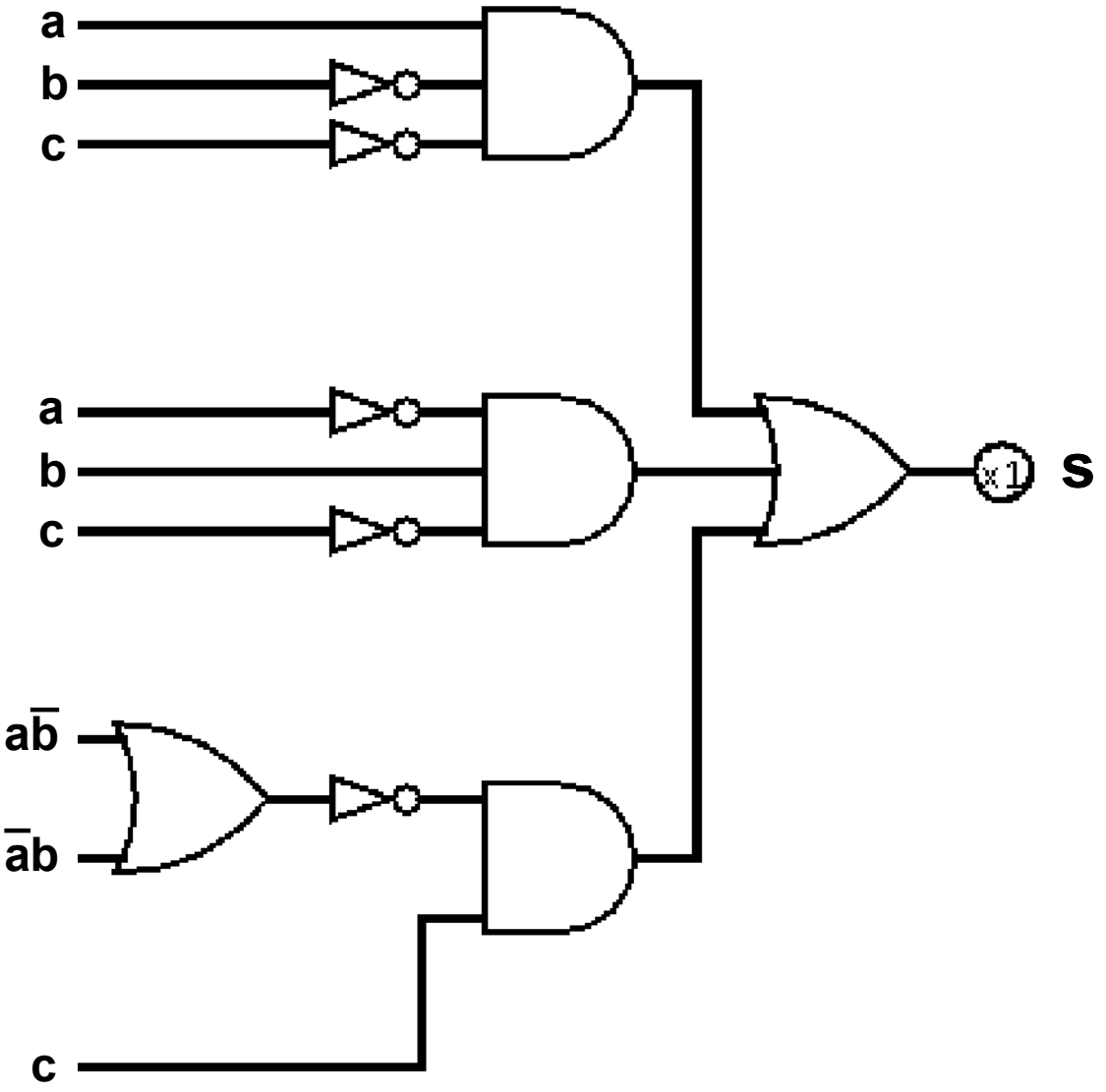
# Boolean Formula → Circuit

$$s = a\bar{b}\bar{c} + \bar{a}b\bar{c} + \sim(a\bar{b} + \bar{a}b)\cdot c$$



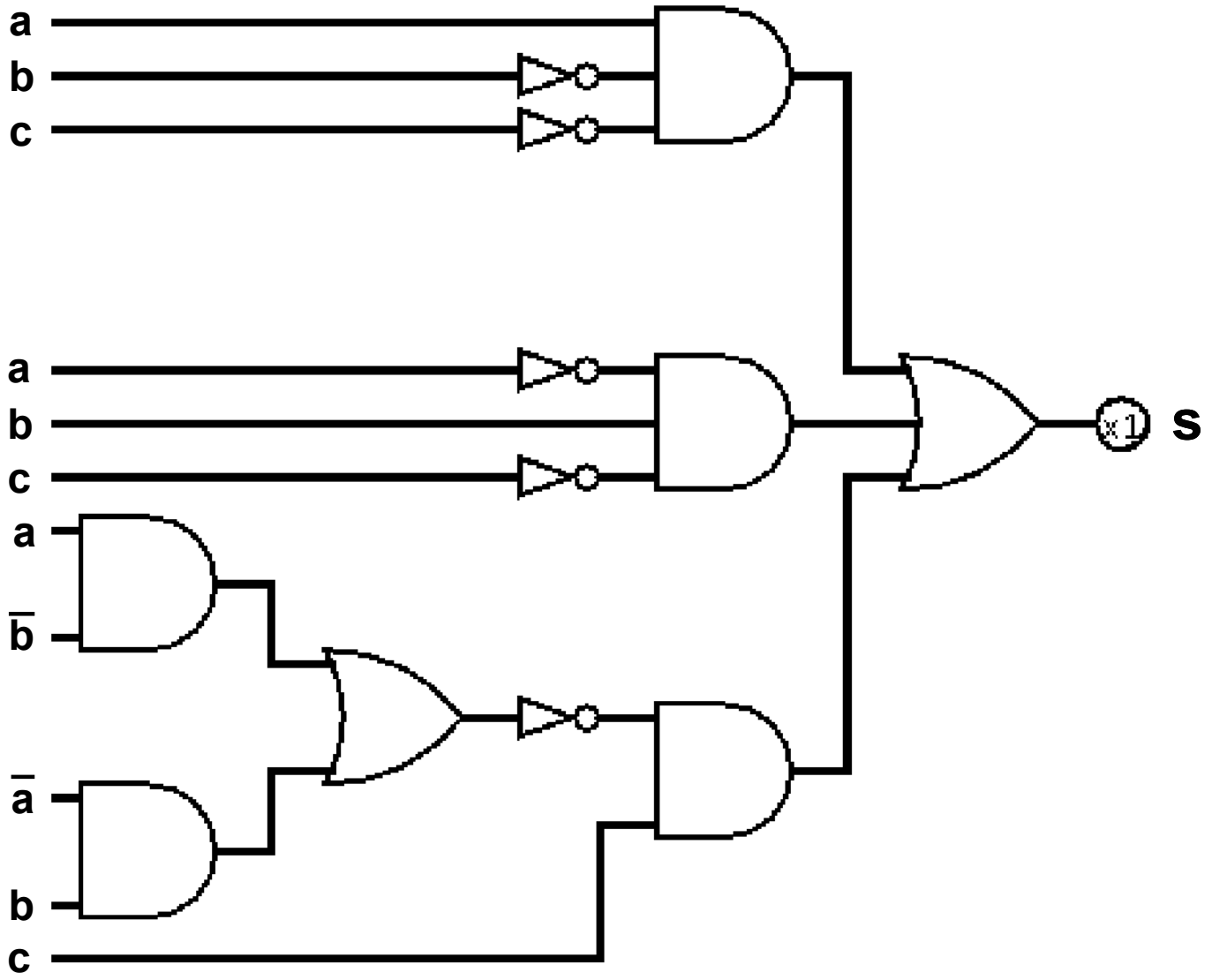
# Boolean Formula → Circuit

$$s = a\bar{b}\bar{c} + \bar{a}b\bar{c} + \sim(a\bar{b} + \bar{a}b)\cdot c$$



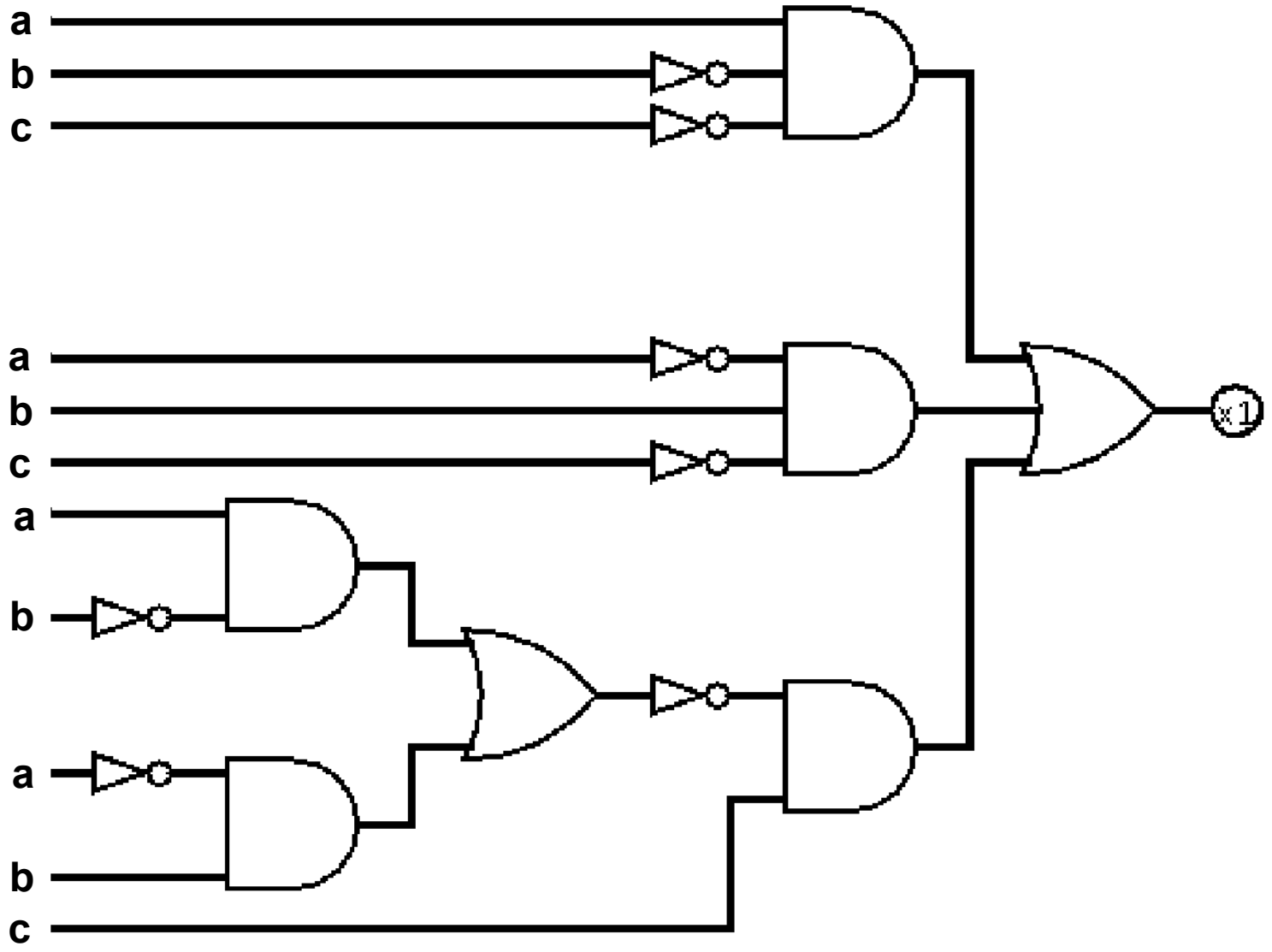
# Boolean Formula → Circuit

$$s = a\bar{b}\bar{c} + \bar{a}b\bar{c} + \sim(a\bar{b} + \bar{a}b)\cdot c$$



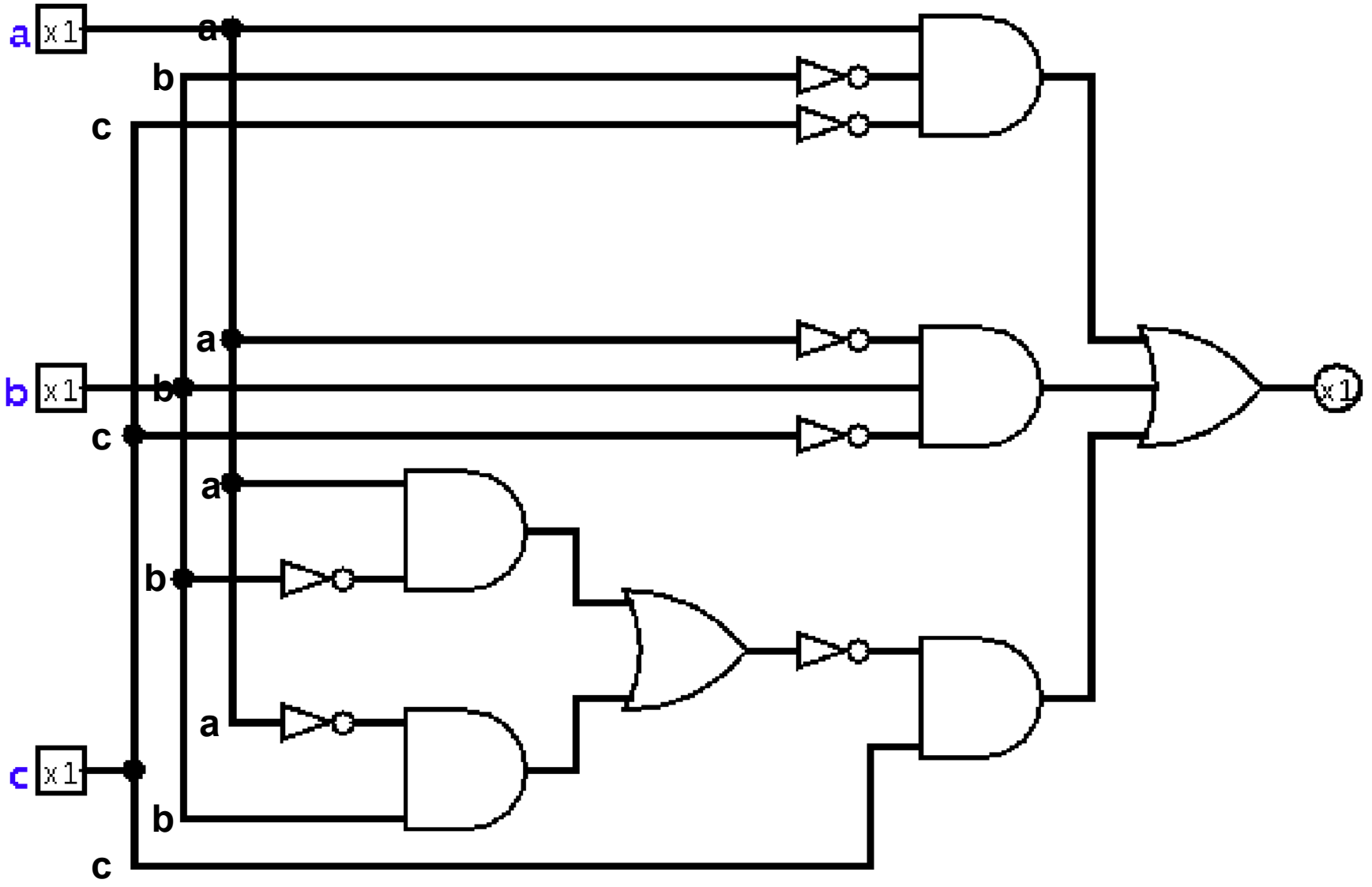
# Boolean Formula → Circuit

$$s = a\bar{b}\bar{c} + \bar{a}b\bar{c} + \sim(a\bar{b} + \bar{a}b)\cdot c$$



# Boolean Formula → Circuit

$$s = a\bar{b}\bar{c} + \bar{a}b\bar{c} + \sim(a\bar{b} + \bar{a}b)\cdot c$$



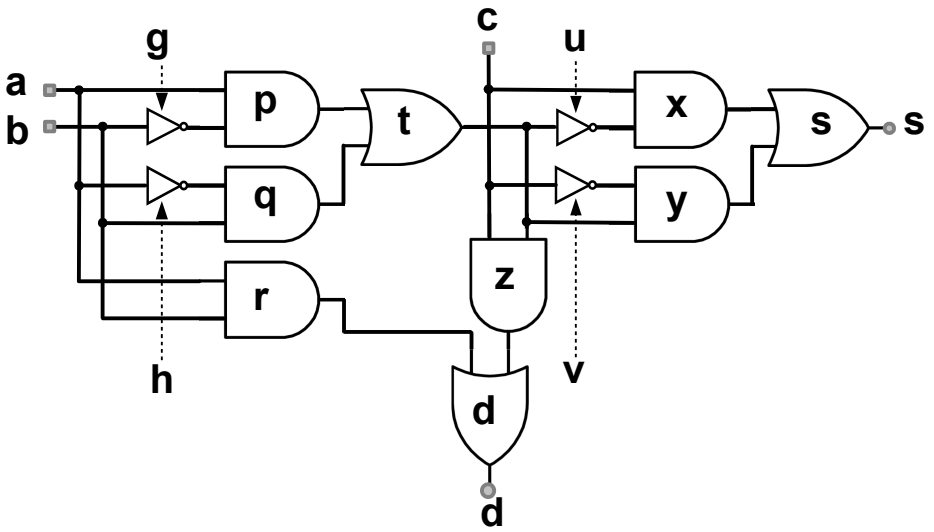
# Boolean Formula → Truth Table

$$s = a\bar{b}\bar{c} + \bar{a}b\bar{c} + \sim(a\bar{b} + \bar{a}b)\cdot c$$

| a | b | c | $\bar{a}$ | $\bar{b}$ | $\bar{c}$ | $a\bar{b}\bar{c}$ | $\bar{a}b\bar{c}$ | $a\bar{b}$ | $\bar{a}b$ | + | ~ | ·c | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | **0** |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | **1** |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | **1** |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | **0** |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | **1** |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | **0** |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | **0** |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | **1** |

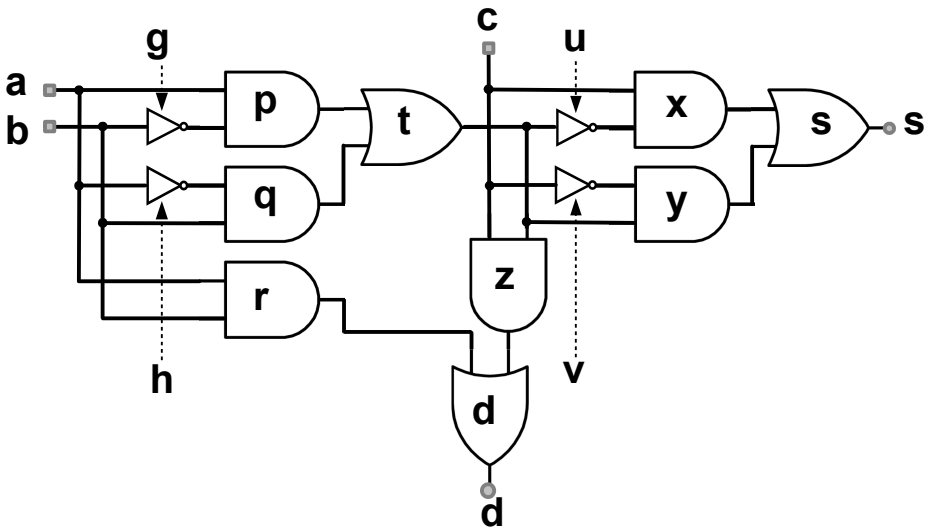
# Formula Simplification

$$
\begin{aligned}
s &= x + y \\
x &= uc \\
y &= tv \\
u &= \bar{t} \\
v &= \bar{c} \\
t &= p + q \\
p &= ag \\
q &= h\bar{b} \\
g &= \bar{b} \\
h &= \bar{a} \\
t &= a\bar{b} + \bar{a}b \\
s &= t\bar{c} + \bar{t}c \\
&= a\bar{b}\bar{c} + \bar{a}b\bar{c} + \bar{t}c \\
&= a\bar{b}\bar{c} + \bar{a}b\bar{c} + \\
&\quad \sim(a\bar{b} + \bar{a}b)\cdot c
\end{aligned}
$$



# Formula Simplification

$$
\begin{aligned}
s &= x + y \\
x &= uc \\
y &= tv \\
u &= \bar{t} \\
v &= \bar{c} \\
t &= p + q \\
p &= ag \\
q &= h\bar{b} \\
g &= \bar{b} \\
h &= \bar{a} \\
t &= a\bar{b} + \bar{a}b \\
s &= t\bar{c} + \bar{t}c \\
&= a\overline{bc} + \bar{a}b\bar{c} + \bar{t}c \\
&= a\overline{bc} + \bar{a}b\bar{c} +
\end{aligned}
$$




$$
\begin{aligned}
\bar{t}c &= \sim(a\bar{b} + \bar{a}b)\cdot c \\
&= \sim(a\bar{b})\cdot\sim(\bar{a}b)\cdot c \\
&= (\bar{a} + b)\cdot(a + \bar{b})\cdot c \\
&= \bar{a}ac + \overline{ab}c + bac + b\bar{b}c \\
&= \overline{ab}c + bac \\
&= abc + \overline{ab}c
\end{aligned}
$$

# Truth Table → Boolean Formula

$$s = a\bar{b}\bar{c} + \bar{a}b\bar{c} + abc + \bar{a}\bar{b}c$$

| a | b | c | $\bar{a}$ | $\bar{b}$ | $\bar{c}$ | $a\bar{b}\bar{c}$ | $\bar{a}b\bar{c}$ | $ab\bar{c}$ | $\bar{a}\bar{b}c$ | + | ~ | ·c | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | **0** |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | **1** |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | **1** |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | **0** |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | **1** |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | **0** |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | **0** |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | **1** |

# Boolean Formula Parse Tree

$X = \overline{S}A + SB$