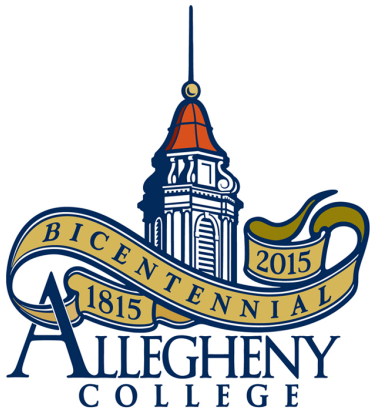
CS200 - Computer Organization

Data Internals - Part2

Aravind Mohan

Allegheny College

October 12, 2021



Motivation to learn data representation

- How is data represented internally?
- How is data represented and processed internally, while executing an operator?

Multiplication and Division

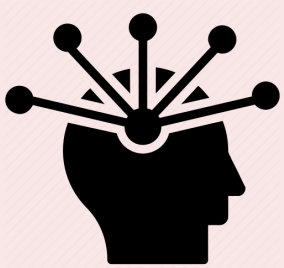
```
1  #include <stdio.h>
2  int main(){
3      int alpha = 10;
4      int beta = 3;
5      int gamma = 5;
6      alpha *= beta;
7      alpha /= gamma;
8      printf("%d\n", alpha);
9  }
```

Q_1: What happens when lines 6, 7,and 8 are executed?

Shift operator

- Shift Left: Makes a number grow bigger.
- Shift Right: Makes a number smaller.
- Test it out: $10 << 2$
- Test it out: $10 >> 2$

Shift operator (c'ntd)

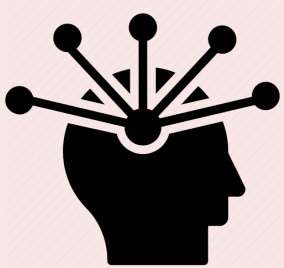


- $\alpha << 0 \implies X \times 1$
- $\alpha << 1 \implies X \times 2$
- $\alpha << 2 \implies X \times 4$
- $\alpha << 3 \implies X \times 8$

α is the binary representation of X

How to represent $X \times 3$ using shift operator?

Shift operator (c'ntd)



- $\alpha >> 0 \implies \frac{X}{1}$
- $\alpha >> 1 \implies \frac{X}{2}$
- $\alpha >> 2 \implies \frac{X}{4}$
- $\alpha >> 3 \implies \frac{X}{8}$

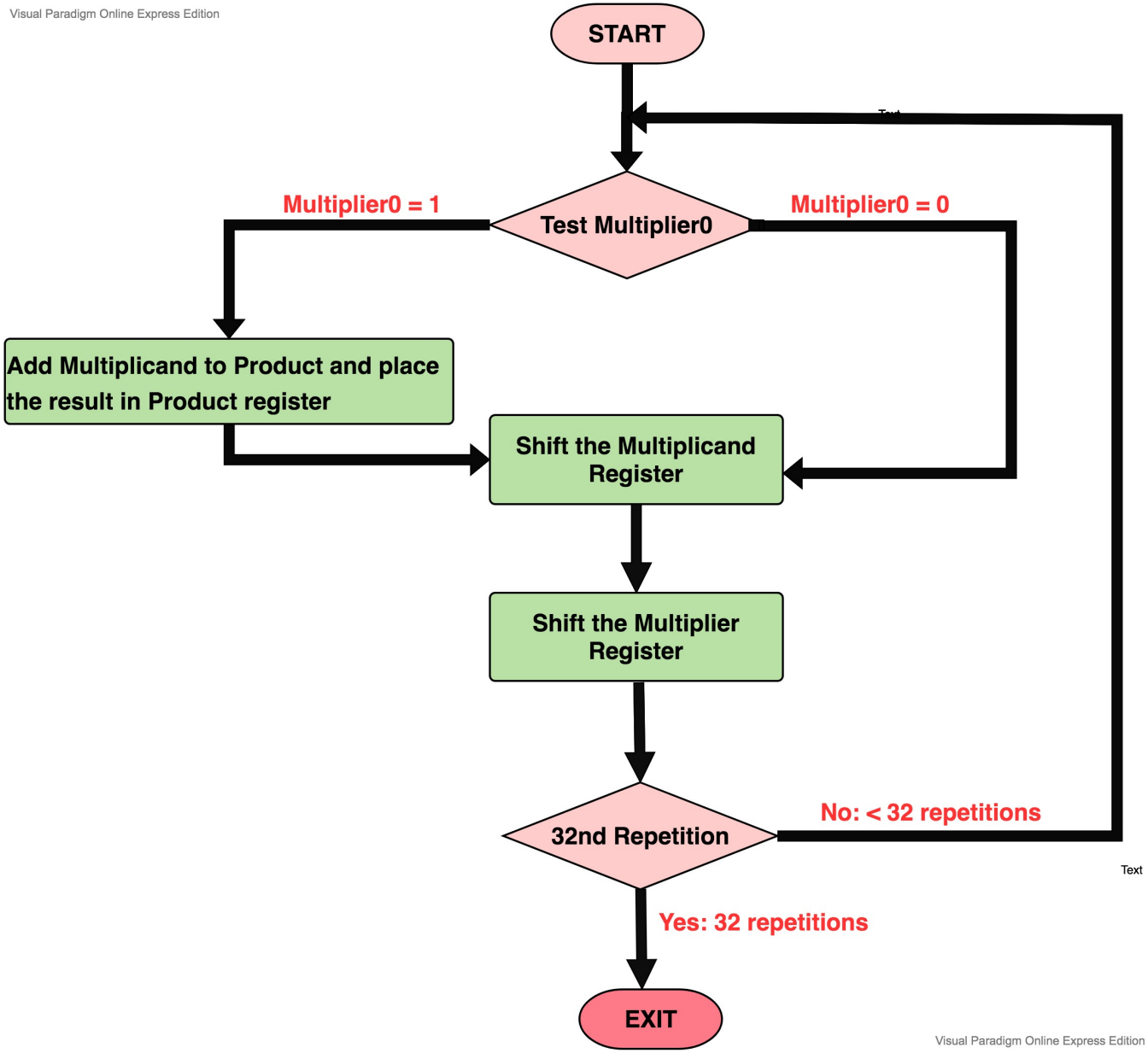
α is the binary representation of X

Multiplication - General Rules

- A process that executes repeated addition.
- Multiplying by any positive number, which is greater than 1, makes the number grow big. Similar to left shift operator.

- Test it out: can we multiply 15 by 2,3,4,5?

Multiplication Algorithm



Multiplication Example - 1

15×7

Multiplicand	Multiplier	Product
0000 1111	0000 0111	0000 0000
←	→	0000 1111
0001 1110	0000 0011	
←	→	0010 1101
0011 1100	0000 0001	
←	→	0110 1001
0111 1000	0000 0000	
←	→	**0110 1001**

Multiplication Example - 2

13×9

Multiplicand	Multiplier	Product
0000 1101	0000 1001	0000 0000
←	→	0000 1101
0001 1010	0000 0100	
←	→	0000 1101
0011 0100	0000 0010	
←	→	0000 1101
0110 1000	0000 0001	
←	→	**0111 0101**

Multiplication Example - 3

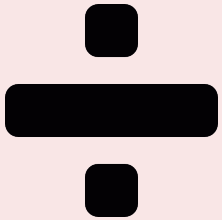
10×13

Multiplicand	Multiplier	Product
0000 1010	0000 1101	0000 0000
←	→	0000 1010
0001 0100	0000 0110	
←	→	0000 1010
0010 1000	0000 0011	
←	→	0011 0010
0101 0000	0000 0001	
←	→	**1000 0010**

Multiplication Try Out

- 5×6
- 7×3

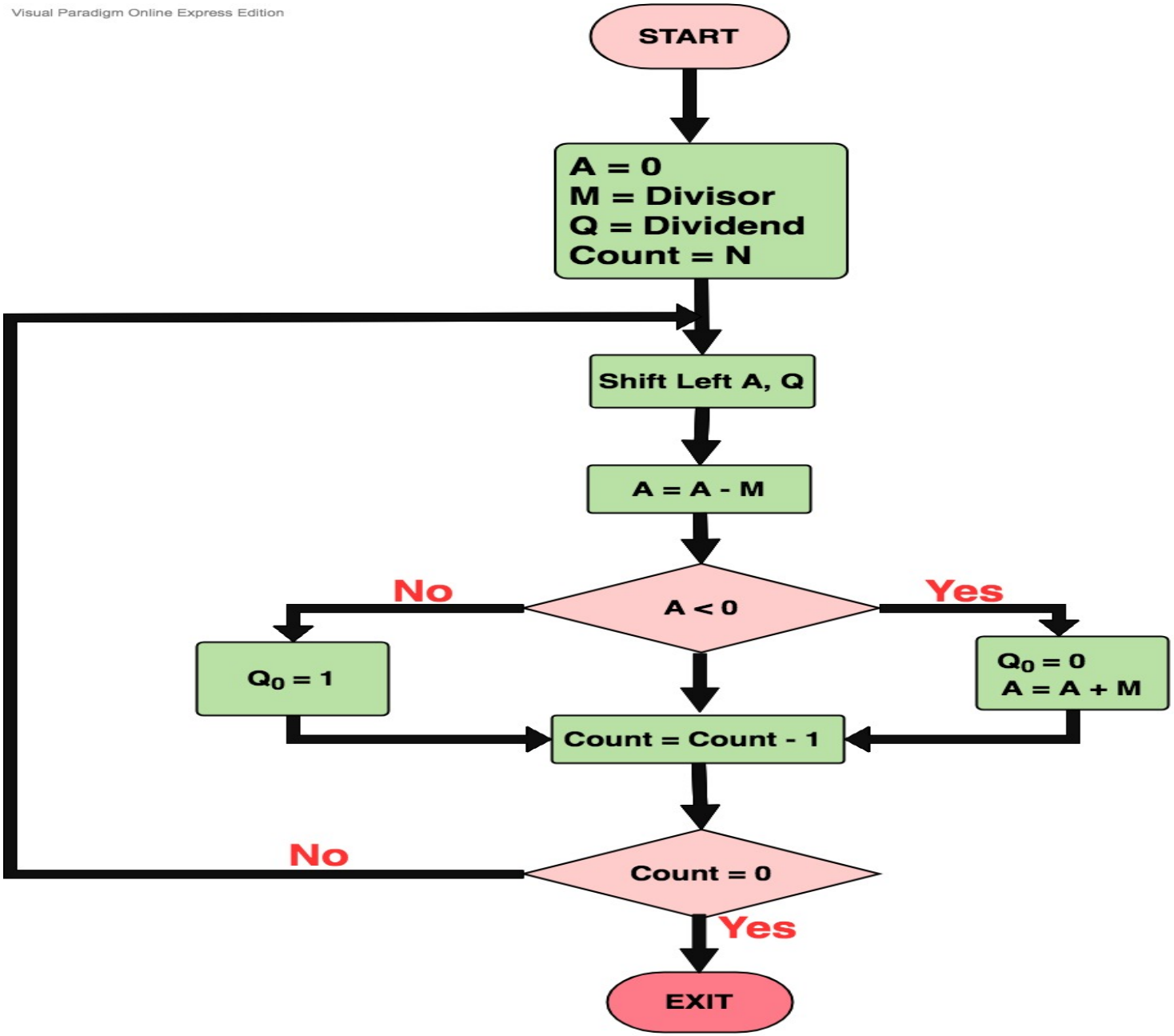
Division - General Rules



- A process that executes repeated subtraction.
- Dividing by any positive number, which is greater than 1, makes the number smaller. Similar to right shift operator.

- Test it out: can we dvide 32 by 2,4,8,15,32?

Division Algorithm



Division Example - 1

$$\frac{12}{5}$$

Operation	A	Q	M	Count
Initialize	00000	1100	00101	4
Left Shift	00001	100-	00101	4
Subtract	00001 + 11011 = **1**1100	1000	00101	4
Sum	11100 + 00101 = 00001	1000	00101	4
	00001	**1000**	**00101**	**3**

Division Example - 1 (contd)

$$\frac{12}{5}$$

Left Shift	00011	000-	00101	3
Subtract	00011 + 11011 = **1**1110	0000	00101	3
Sum	00011	0000	00101	3
	00011	**0000**	**00101**	**2**
Left Shift	00110	000-	00101	2
Subtract	00110 + 11011 = 00001	0001	00101	2
	00001	**0001`**	**00101**	**1**

Division Example - 1 (contd)

$$\frac{12}{5}$$

Left Shift	00010	001-	00101	1
Subtract	00010 + 11011 = 11101	0010	00101	1
Sum	00010	0010	00101	1
	00010	**0010**	**00101**	**0**

Let us try out some examples?

- $10/2$
- $13/5$
- $16/7$
- $17/6$

Next ...

- Logic gates
- Assembly language programming

Reading Assignment

- **PH** - chapter 03: [3.3];

Questions

Do you have any questions from this class discussion?