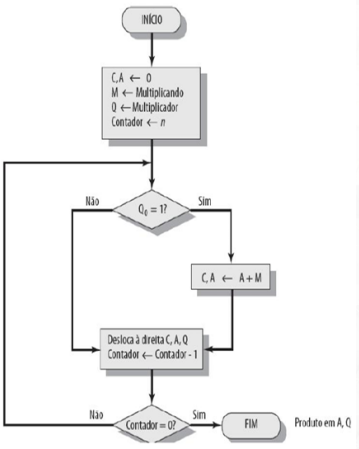
Faça os seguintes produtos utilizando o algoritmo de multiplicação de inteiros sem sinal apresentado em aula. Apresente os resultados apresentando cada passo do algoritmo. Considere números de 4 bits.

a) 3 x 3

b) 4 x 2

c) 8 x 4



a) 3 × 3 → M = 0011 × Q = 0011

A	Q	
0000	0011	n=4 Soma A+M
0011	0011	Desloca dir n=3
0001	1001	Soma A+M
0100	1001	Desloca dir n=2
0010	0100	Desloca dir n=1
0001	0010	Desloca dir n=0
0000	1001	⇒ 9 ✓

b) 4 × 2 → M = 0100 × Q = 0010

A	Q	
		n=4
0000	0010	Desloca dir n=3
0000	0001	Soma A+M
0100	0001	Desloca dir n=2
0010	0000	Desloca dir n=1
0001	0000	Desloca dir n=0
0000	1000 → 8 ✓	

c) 8 × 4 → M = 1000 × Q = 0100

n=4

A	Q	
0000	0100	desloca dir n=3
0000	0010	desloca dir n=2
0000	0001	Soma A+M
1000	0001	desloca dir n=1
0100	0000	desloca dir n=0
0010	0000 → 32 ✓	

32 16 8 4 2 1

a) -3 x -5

b) 7 x 9

c) 9 / 3

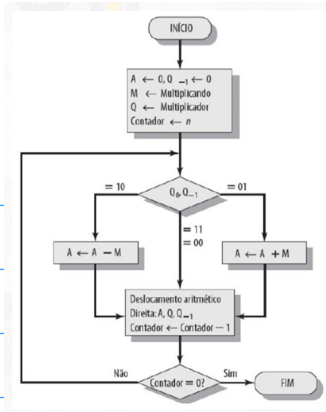
d) 8 / 5

$3 \rightarrow 0011 \rightarrow -3 = 1101$

$5 \rightarrow 0101 \rightarrow -5 = 1011$

$1101 - 1011 = 0010$

$1110 - 1011 = 0011$



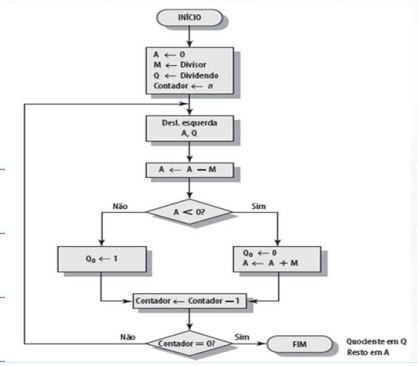
a) $-3 \times -5 \rightarrow 1101 \times 1011$ (Q, M)

A	Q	Q-1	n = 4
0000	1101	0	A − M
0101	1101	0	Desloc dir Aritmetico (n = 3)
0010	1110	1	A + M
1101	1110	1	Desloca dir Aritimetico (n = 2)
1110	1111	0	A − M
0011	1111	0	Desloca dir Aritimetico (n = 1)
0001	1111	1	Desloca (n = 0)
0000	1111	1	→ 15 ✓

b) $7 \times 9 \rightarrow 0111 \times 1001$ (Q, M)

A	Q	n = 4
0000	0111	Soma A + M
1001	0111	desloca dir (n = 3)
0100	1011	Soma A + M
1101	1011	Desloca dir (n = 2)
0110	1101	Soma A + M
1111	1101	desloca dir (n = 1)
0111	1110	Desloca dir (n = 0)
0011	1111	→ 63 ✓

0010 | 0001
1110 | 1111



c) 9/3 → Q: 1001 / M: 0011

	A	Q	
	0000	1001	n=4 Desloc Esq
	0001	0010	A − M
-2	1110	0010	$Q_0=0$ A+M n=3
	0001	0010	Desloc Esq
	0010	0100	A − M
-1	1111	0100	$Q_0=0$ A+M n=2
	0010	0100	Desloc Esq
	0100	1000	A − M
	0001	1000	$Q_0=1$ n=1
	0001	1001	Desloc Esq
	0011	0010	A − M
	0000	0010	$Q_0=1$ n=0
	0000	0011	→ 3 ✓

- **Deslocamento Lógico**
  - Entra 0
- **Deslocamento Aritmético**
  - entra o último bit deslocado

- **Num Negativo**
  - Localiza o 1 mais a direita, mantem ele e inverte os outros a esquerda dele

Datapath

Clock monociclo

DESVANTAGENS

- É limitado pela instrução de maior tempo
- Menos otimizado caso tenha instruções que executem em tempos diferentes (Intervalos de)

VANTAGEM

- Simplicidade na implementação

Instruções que usam extensora de sinais

- Basicamente todas com imediato
- ADDI, SUBI...
- Load/store (memória)
- Breach (Beq, BNe)

⇨ Precisa do Extensor, pois o processador trabalha somente com 32 bits, então é necessário estender o imediato completando-o com 1 ou 0 dependendo do sinal para que tenha 32 bits.

ULA do Lw e Sw ⇨ Reg + Offset

Usa o add para somar o valor do registrador com o offset para obter o endereço de memória que se quer.

Caminho de dados

- Instruções Normais

PC → Instruction memory → banco de Regs → ULA → Wb

- Com memória (LW/SW)

PC → Instruction mem → banco de Regs → Extensor do sinal → ULA → Data mem → Wb

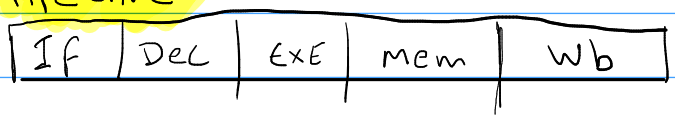
- J

PC → Instruction mem → CALCULO do NOVO PC → PC (control, MUX)

- Beq

PC → Instruction mem → BR → Extensor de sinal + Shift Left 2 → ULA → comparação (MUX) → PC END PROX INSTRUÇÃO

Pipeline



* bolha
  - Usado no método sem forwarding

* Com forwarding
  - O dado já fica disponível no EXE para evitar conflito.
  - Lw / Sw precisam de bolha mesmo assim, pois o dado só ficam disponível no WB.

* flush
  - "Reseta" o pipeline caso haja um desvio ou salto.

A partir das seguintes instruções, simule a execução do **pipeline** do mips por meio de um teste de mesa.

```
add $t0,$zero,$t1
lw $s2,0($sp)
addi $t4, $s2, 10
sll $s0,$s1, 2
```

Como é Lw não faz diferença se usa forwarding ou não

	IF	Dec	EXE	mem	WB
1	ADD				
2	Lw	ADD			
3	Bolha	Lw	ADD		
4	ADDi	Bolha	Lw		ADD
5	SLL	ADDi	Bolha	Lw	
6		SLL	ADDi	Bolha	Lw
7			SLL		ADDi
8					SLL
9					

A partir das seguintes instruções, simule a execução do **pipeline** do mips por meio de um teste de mesa.

```
add $t0,$zero,$t1
sub $t2,$t2,$t3
addi $t4, $t4, 10
sll $s0,$s1, 2
lw $s2,0($sp)
```

	If	Dec	ExE	MEM	WB
1	Add				
2	Sub	Add			
3	Addi	Sub	Add		
4	Sll	Addi	Sub		Add
5	Lw	Sll	Addi		Sub
6		Lw	Sll		Addi
7			Lw		Sll
8				Lw	
9					Lw

A partir das seguintes instruções, simule a execução do **pipeline** do mips por meio de um teste de mesa.

```
add $t0,$zero,$t1
lw $s2,0($sp)
addi $t4, $t4, 10
sll $s0,$s1, 2
```

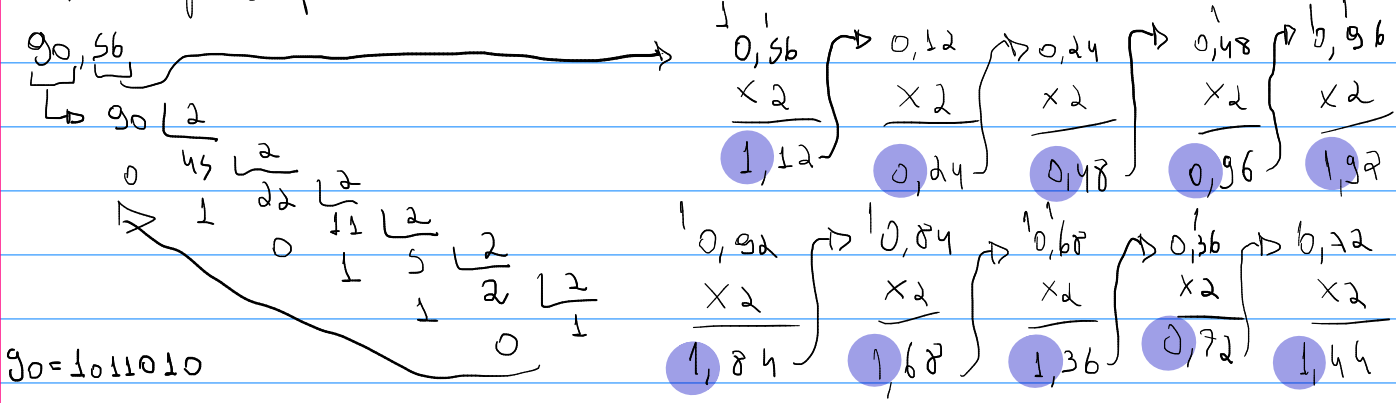
	If	Dec	EXE	mem	WB
1	ADD				
2	Lw	ADD			
3	ADDi	Lw	ADD		
4	Sll	ADDi	Lw		ADD
5		Sll	ADDi	Lw	
6		Sll	ADDi	Bolinha	Lw
7			Sll		ADDi
8					Sll

A partir das seguintes instruções, simule a execução do **pipeline** do mips por meio de um teste de mesa.

```
add $t0,$zero,$t1
lw $s2,0($sp)
addi $t4, $s2, 10
sll $s0,$s1, 2
```

	If	Dec	EXE	mem	WB
1	ADD				
2	LW	ADD			
3	Bolha	LW	ADD		
4	ADDi	Bolha	LW		ADD
5	SLL	ADDi	Bolha	LW	LW
6		SLL	ADDi		
7			SLL		ADDi
8					SLL

=> Formato ponto flutuante



90=1011010

0,56=1000111101

90,56 = 1011010,1000111101

ou 1,011010100011110 1.2^6 ->

1 8 23

SINAL BASE MATISSA

0 10000101 01101010001111010000000

127+6 = 133 | 2
1 66 | 2
0 33 | 2
1 16 | 2
0 8 | 2
0 4 | 2
0 2 | 2
0 1

10000101

Soma de inteiros com ponto flutuante

PASSO 1º Normalização dos números

PASSO 2º Alinhar os mantissas e deixar os expoentes iguais

PASSO 3º Soma das mantissas

PASSO 4º Normalizar o resultado, deslocar os dígitos da mantissa até que o mais significativo seja $\neq 0$.

Soma bin	Subtração bin
$\frac{1}{1} = 1$, $\frac{1}{0} = 1$	$\frac{1}{1} = 0$, $\frac{-1}{0} = 1$
$\frac{0}{0} = 0$, $\frac{0}{1} = 1$	$\frac{0}{0} = 0$, $\frac{0}{1} = -0$