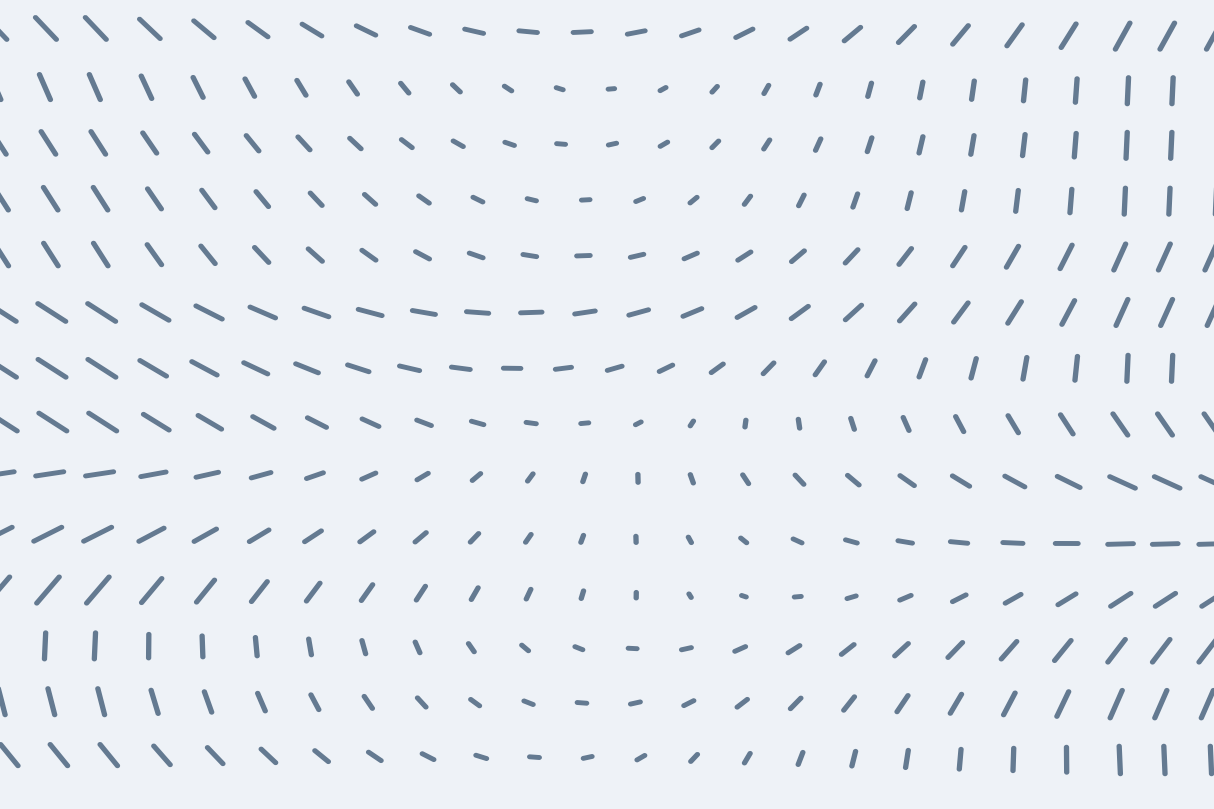
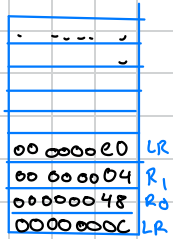


Stack

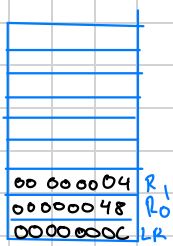
demo en el video

iteración n-2

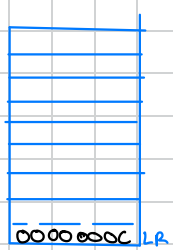


https://drive.google.com/drive/folders/1EYcnWcQHsoRjI-fqdlvCv_UEOs6X6nkx?usp=drive_link

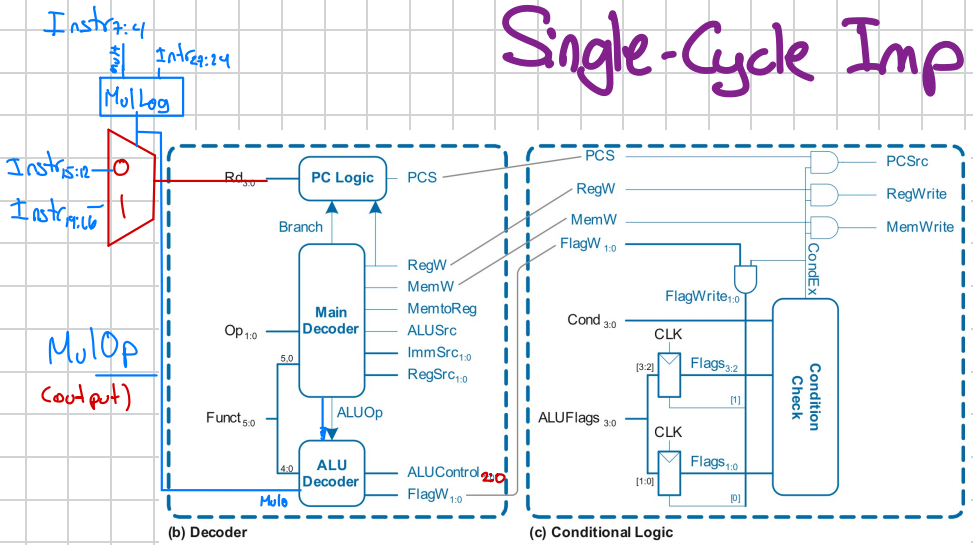
iteración n-1

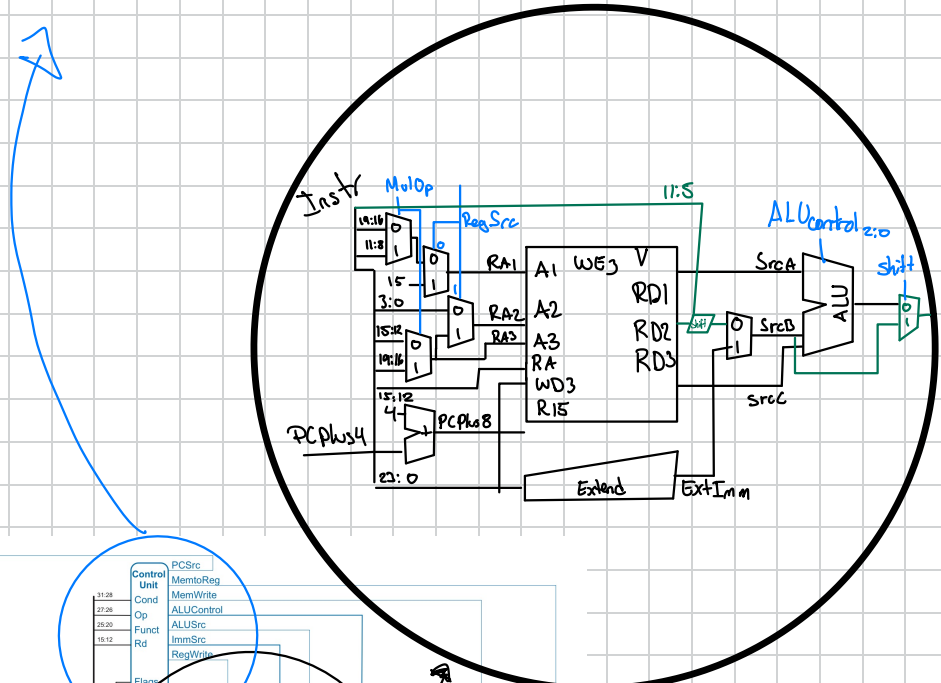


iteración n



Single-Cycle Impl.





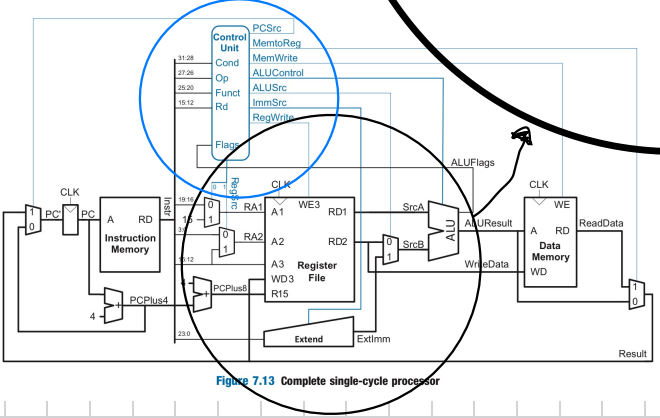
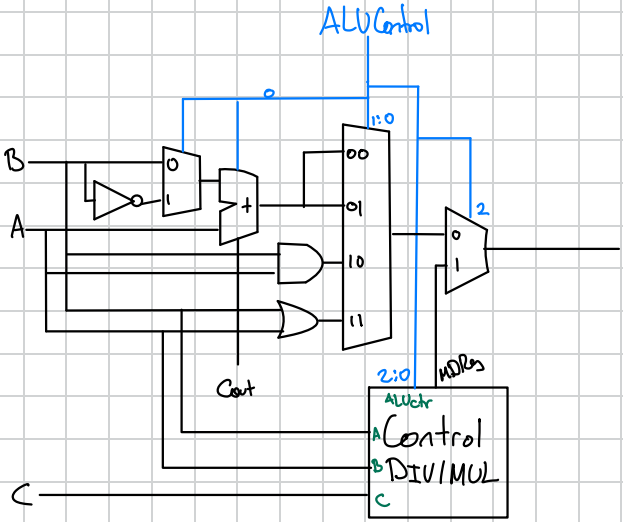
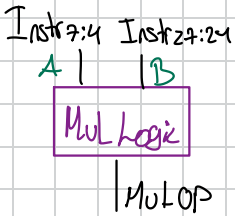


Figure 7.13 Complete single-cycle processor

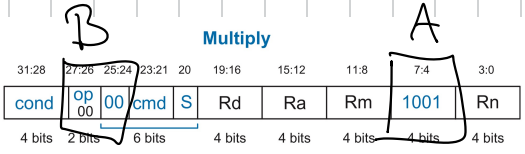
ALU:



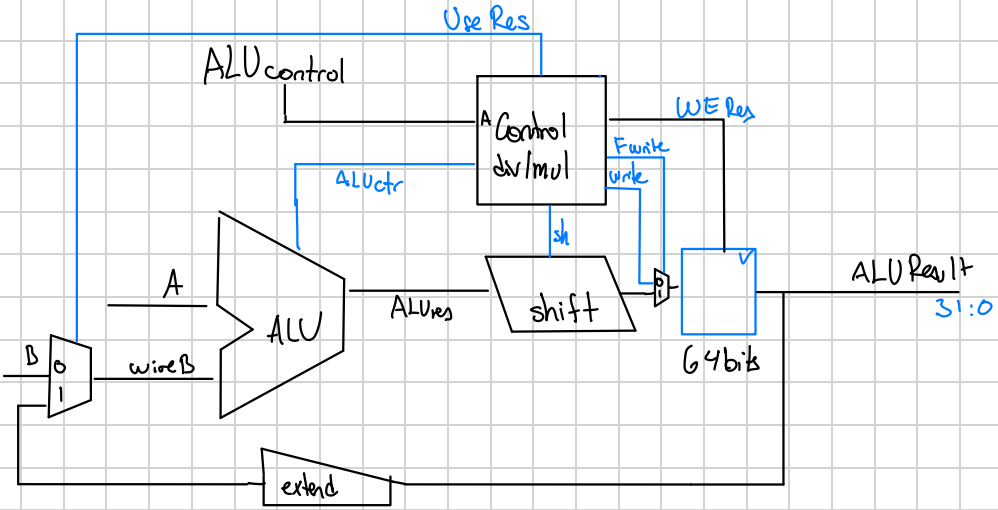
MULLogic:



$\sim B \& A == 1001$



Control DIV/MUL



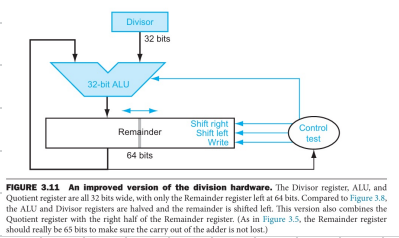


FIGURE 3.11 An improved version of the division hardware. The Divisor register, ALU, and Quotient register are all 32 bits wide, with only the Remainder register left at 64 bits. Compared to Figure 3.8, the ALU and Divisor registers are halved and the remainder is shifted left. This version also combines the Quotient register with the right half of the Remainder register. (As in Figure 3.5, the Remainder register should really be 65 bits to make sure the carry out of the adder is not lost.)

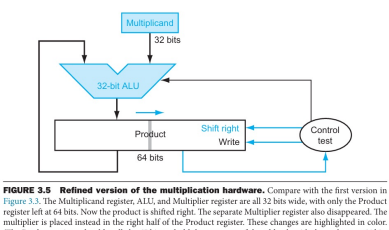


FIGURE 3.5 Refined version of the multiplication hardware. Compare with the first version in Figure 3.3. The Multiplicand register, ALU, and Multiplier register are all 32 bits wide, with only the Product register left at 64 bits. Now the product is shifted right. The separate Multiplier register also disappeared. The multiplier is placed instead in the right half of the Product register. These changes are highlighted in color. (The Product register should really be 65 bits to hold the carry out of the adder, but it's shown here as 64 bits to highlight the evolution from Figure 3.3.)

Recuperado de: (bibliografía complementaria del curso)

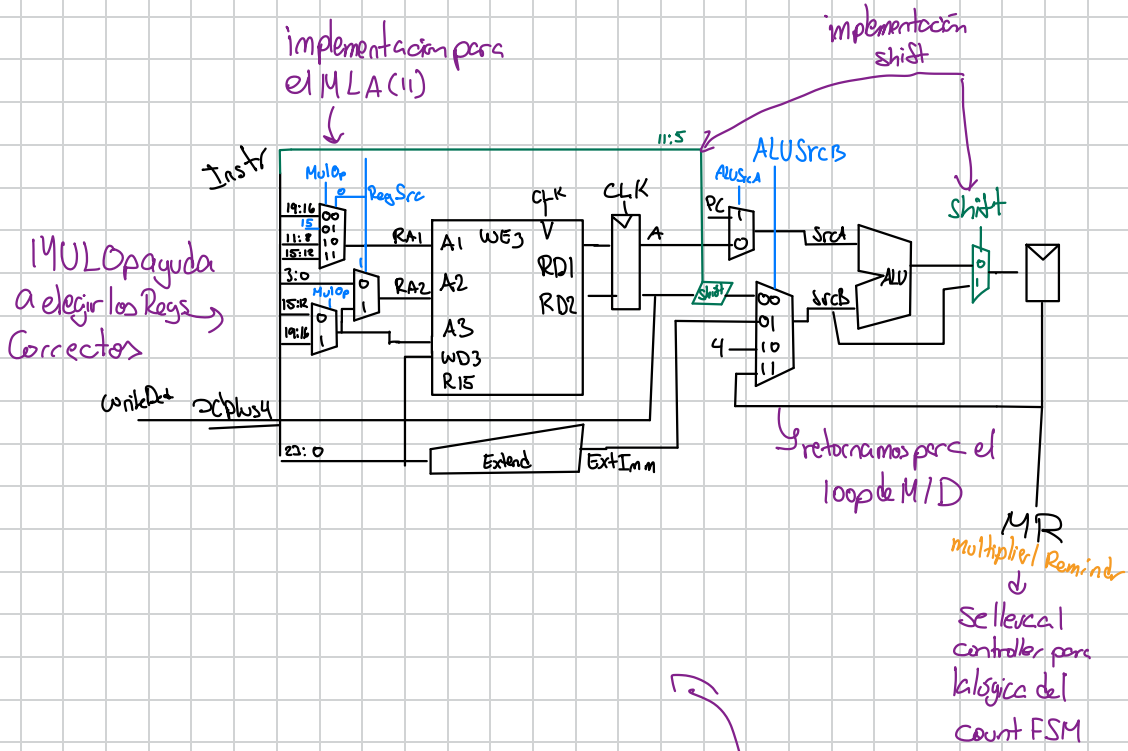
Patterson, D. A., & Hennessy, J. L. (2016). Computer Organization andDesign ARM Edition: The Hardware Software Interface. Morgan Kaufmann.

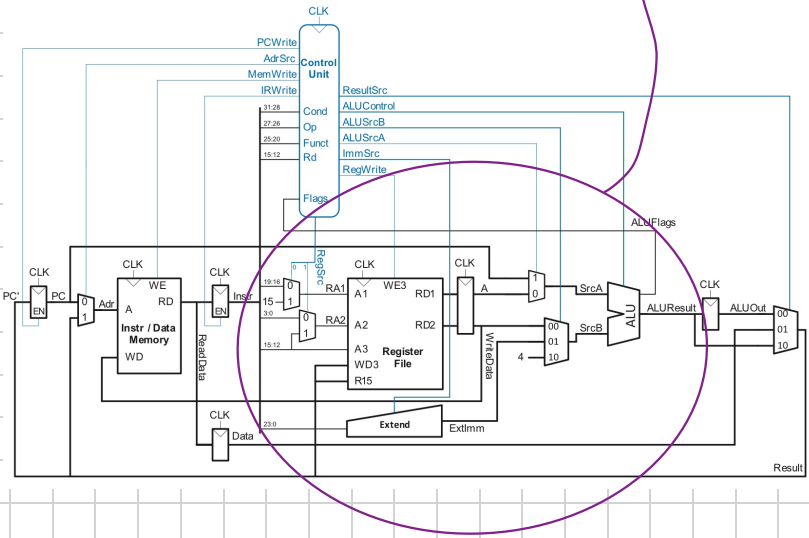
CMD

001	MLA
100	SDIV
110	UDIV

↓ divide ↓ accumulate

Implementación Multi-cycle





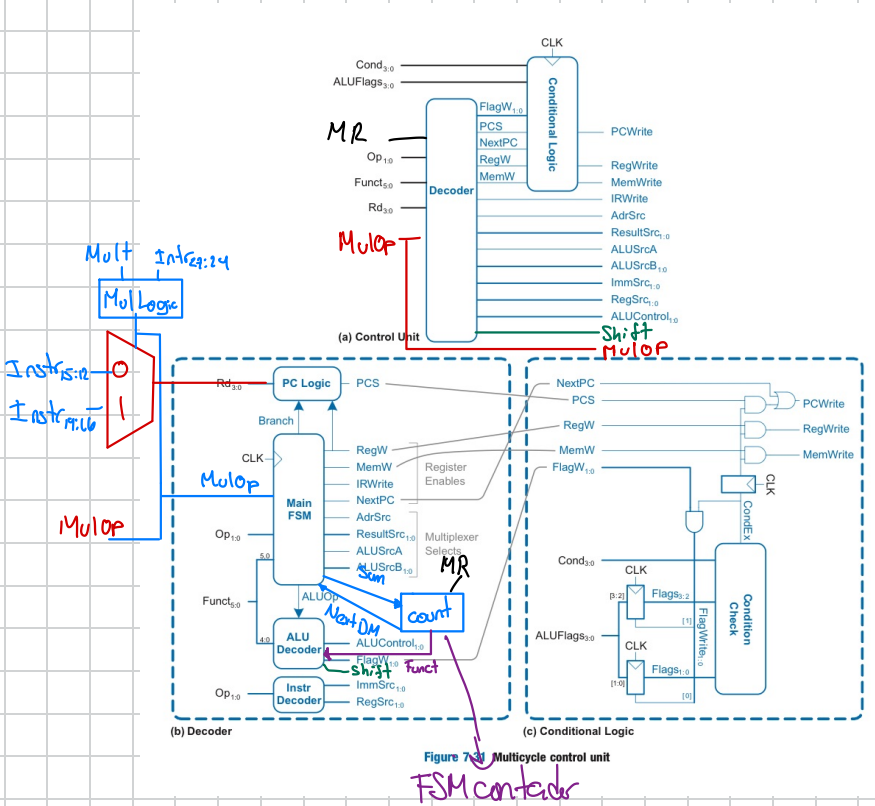
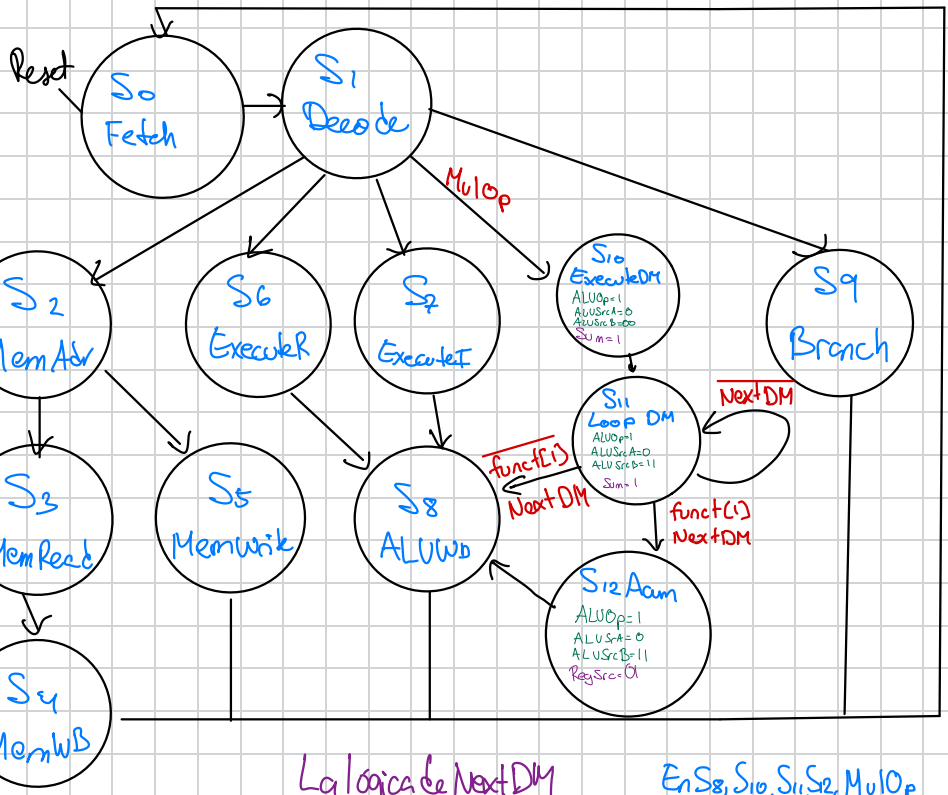


Figure 7.31 Multicycle control unit

FSM controller



La lógica de NextDM se encarga el FSM Contado

En S8, S10, S1, S2, MulOp va a estar en 1, los regs se leen y guardan bien

S10: ingresa contenido de ambos registros

S11: empieza a tomar en SrcB, el anterior valor, SrcA queda su valor anterior

S12: en caso se necesite MLA, se toma el RA y se suma a la multiplicación obtenida

State (Name)	NextPC	Branch	MemW	RegW	IRWrite	AdrSrc	ResultSrc1:0	ALUSrcA1:0	ALUSrcB1:0	ALUOp	Sum
0 (Fetch)	1	0	0	0	1	0	10	0	1	10	0	0
1 (Decode)	0	0	0	0	0	0	10	0	1	10	0	0
2 (MemAdr)	0	0	0	0	0	0	00	0	0	01	0	0
3 (MemRead)	0	0	0	0	0	1	00	0	0	00	0	0
4 (MemWB)	0	0	0	1	0	0	01	0	0	00	0	0
5 (MemWrite)	0	0	1	0	0	1	00	0	0	00	0	0
6 (ExecuteR)	0	0	0	0	0	0	00	0	0	00	1	0
7 (ExecuteI)	0	0	0	0	0	0	00	0	0	01	1	0
8 (ALUWB)	0	0	0	1	0	0	00	0	0	00	0	0
9 (Branch)	0	1	0	0	0	0	10	1	0	01	0	0
10(ExeuteDM)	0	0	0	0	0	0	00	0	0	00	1	1
11 (LoopDM)	0	0	0	0	0	0	00	0	0	11	1	1
12 (Mcum)	0	0	0	0	0	0	01	0	0	11	1	0

Count FSM

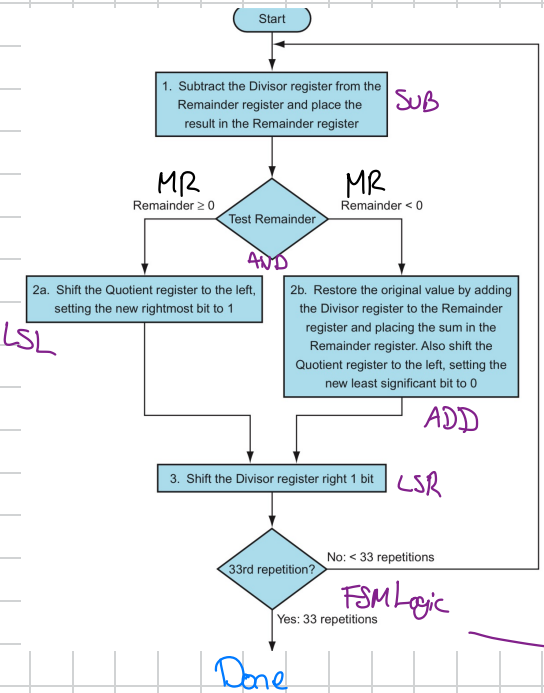
FSM que toma la lógica de MLA, SDIV, UDIV

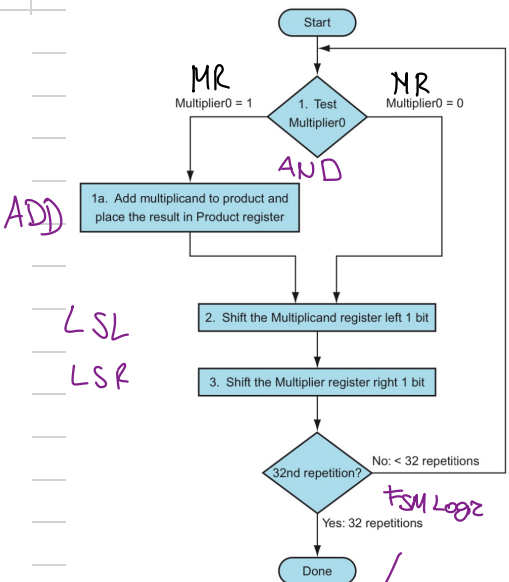
CMD

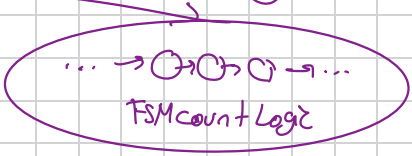
001	MLA
100	SDIV
110	UDIV

↙ divide ↓ accumulate

Diagrama de flujo:







Recuperado de: (bibliografía complementaria del curso)

Patterson, D. A., & Hennessy, J. L. (2016). Computer Organization andDesign ARM Edition: The Hardware Software Interface. Morgan Kaufmann.