

Máquina de Estados Finitos

Unidade de Controle

Base e Extensões

Base	Descrição	Versão	Congelada?
RV32I	Conjunto de Instruções com Inteiros	2.0	Sim
RV32E	Conjunto de Instruções com Inteiros (sistemas embarcados)	1.9	Não
RV64I	Conjunto de Instruções com Inteiros	2.0	Sim
RV128I	Conjunto de Instruções com Inteiros	1.7	Não

Base e Extensões

Base	Descrição
RV32I	Conjunto de Instruções com Inteiros
RV32E	Conjunto de Instruções com Inteiros (si
RV64I	Conjunto de Instruções com Inteiros
RV128I	Conjunto de Instruções com Inteiros

Extensão	Descrição	Versão	Congelada?
M	Extensão para Multiplicação e Divisão de Inteiros	2.0	Sim
A	Extensão para Instruções Atômicas	2.0	Sim
F	Extensão para Ponto Flutuante de Precisão Simples	2.0	Sim
D	Extensão para Ponto Flutuante de Precisão Dupla	2.0	Sim
Q	Extensão para Ponto Flutuante de Precisão Quadrupla	2.0	Sim
L	Extensão para Ponto Flutuante Decimal	0.0	Não
C	Extensão para Instruções Compactas	2.0	Sim
B	Extensão para Manipulação de Bits	0.0	Não
J	Extensão para Linguagens Dinamicamente Traduzidas	0.0	Não
T	Extensão para Memória Transacional	0.0	Não
P	Extensão para Instruções SIMD	0.1	Não
V	Extensão para Operações Vetoriais	0.2	Não
N	Extensão para Interrupções a nível de Usuário	1.1	Não

Formato das instruções - RV32I

31	30	25	24	21	20	19	15	14	12	11	8	7	6	0			
funct7				rs2			rs1		funct3		rd			opcode		Tipo R	
imm[11:0]						rs1		funct3		rd			opcode		Tipo I		
imm[11:5]				rs2			rs1		funct3		imm[4:0]			opcode		Tipo S	
imm[12]		imm[10:5]		rs2			rs1		funct3		imm[4:1]		imm[11]		opcode		Tipo B
imm[31:12]										rd			opcode		Tipo U		
imm[20]		imm[10:1]			imm[11]		imm[19:12]			rd			opcode		Tipo J		

Registradores

31	x0 / zero	Zero hardwired
	x1 / ra	Endereço de retorno
	x2 / sp	Ponteiro de pilha
	x3 / gp	Ponteiro global
	x4 / tp	Ponteiro de thread
	x5 / t0	Temporário
	x6 / t1	Temporário
	x7 / t2	Temporário
	x8 / s0 / fp	Registrador salvo, ponteiro de quadro
	x9 / s1	Registrador salvo
	x10 / a0	Argumento de função, valor de retorno
	x11 / a1	Argumento de função, valor de retorno
	x12 / a2	Argumento de função
	x13 / a3	Argumento de função
	x14 / a4	Argumento de função
	x15 / a5	Argumento de função
	x16 / a6	Argumento de função
	x17 / a7	Argumento de função
	x18 / s2	Registrador salvo
	x19 / s3	Registrador salvo
	x20 / s4	Registrador salvo
	x21 / s5	Registrador salvo
	x22 / s6	Registrador salvo
	x23 / s7	Registrador salvo
	x24 / s8	Registrador salvo
	x25 / s9	Registrador salvo
	x26 / s10	Registrador salvo
	x27 / s11	Registrador salvo
	x28 / t3	Temporário
	x29 / t4	Temporário
	x30 / t5	Temporário
	x31 / t6	Temporário

RV32C: Instruções Compactadas

Formato	Significado	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
CR	Registrador	funct4				rd/rs1				rs2				op					
CI	Imediato	funct3		imm		rd/rs1				imm				op					
CSS	Store relativo a pilha	funct3		imm						rs2				op					
CIW		funct3		imm								rd'		op					
CL	Load	funct3		imm			rs1'		imm		rd'		op						
CS	Store	funct3		imm			rs1'		imm		rs2'		op						
CB	Desvio	funct3		offset			rs1'		offset				op						
CJ	Salto	funct3		jump target												op			

RVC Register Number

Integer Register Number

Integer Register ABI Name

000	001	010	011	100	101	110	111
x8	x9	x10	x11	x12	x13	x14	x15
s0	s1	a0	a1	a2	a3	a4	a5

Opcode 00

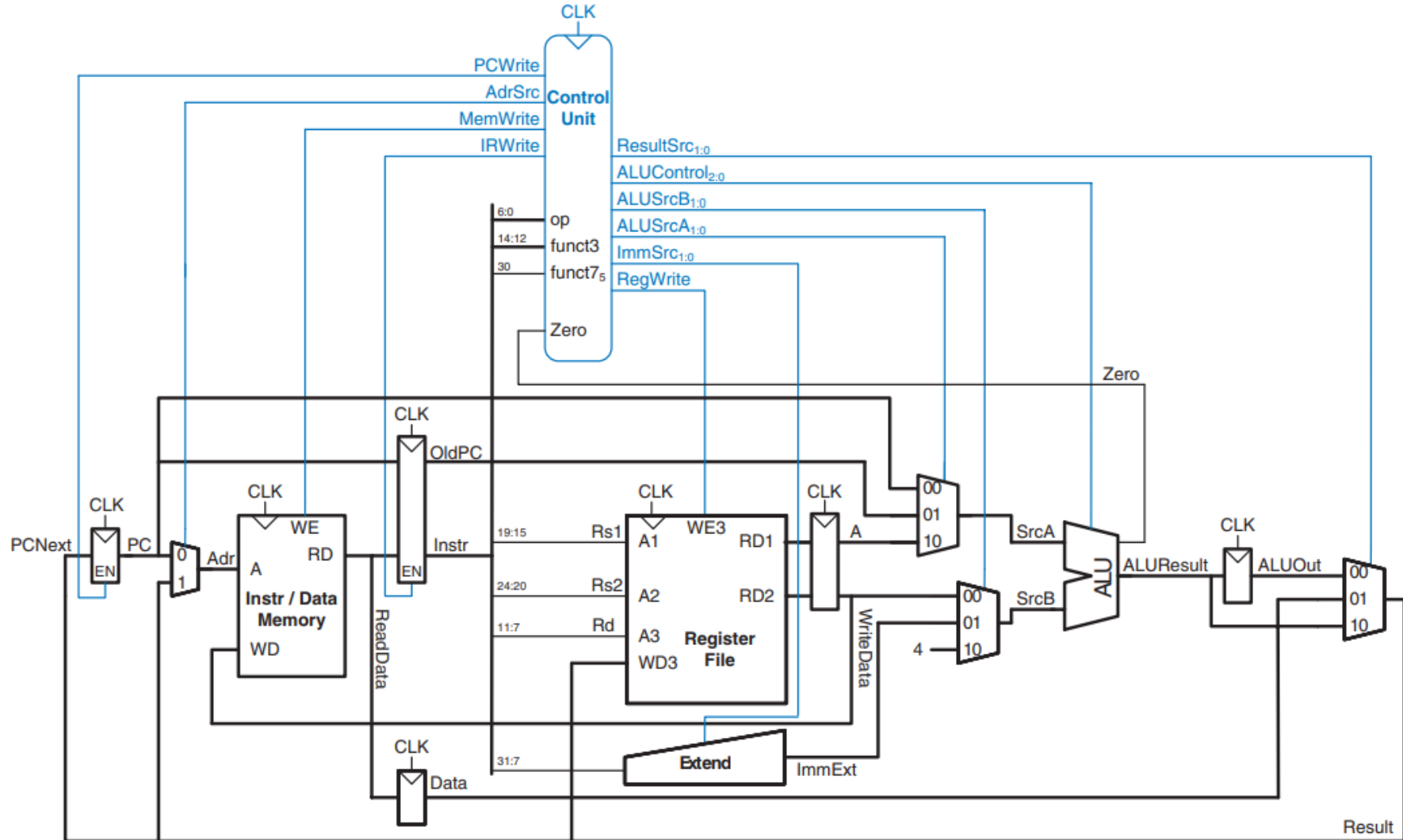
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
000			0								0		00		CIW <i>Illegal instruction</i>	
000			nzuimm[5:4 9:6 2 3]								rd'		00		CIW c.addi4spn	
001			uimm[5:3]			rs1'			uimm[7:6]			rd'		00		CL c.fld
010			uimm[5:3]			rs1'			uimm[2:6]			rd'		00		CL c.lw
011			uimm[5:3]			rs1'			uimm[2:6]			rd'		00		CL c.flw
101			uimm[5:3]			rs1'			uimm[7:6]			rs2'		00		CL c.fsd
110			uimm[5:3]			rs1'			uimm[2:6]			rs2'		00		CL c.sw
111			uimm[5:3]			rs1'			uimm[2:6]			rs2'		00		CL c.fsw

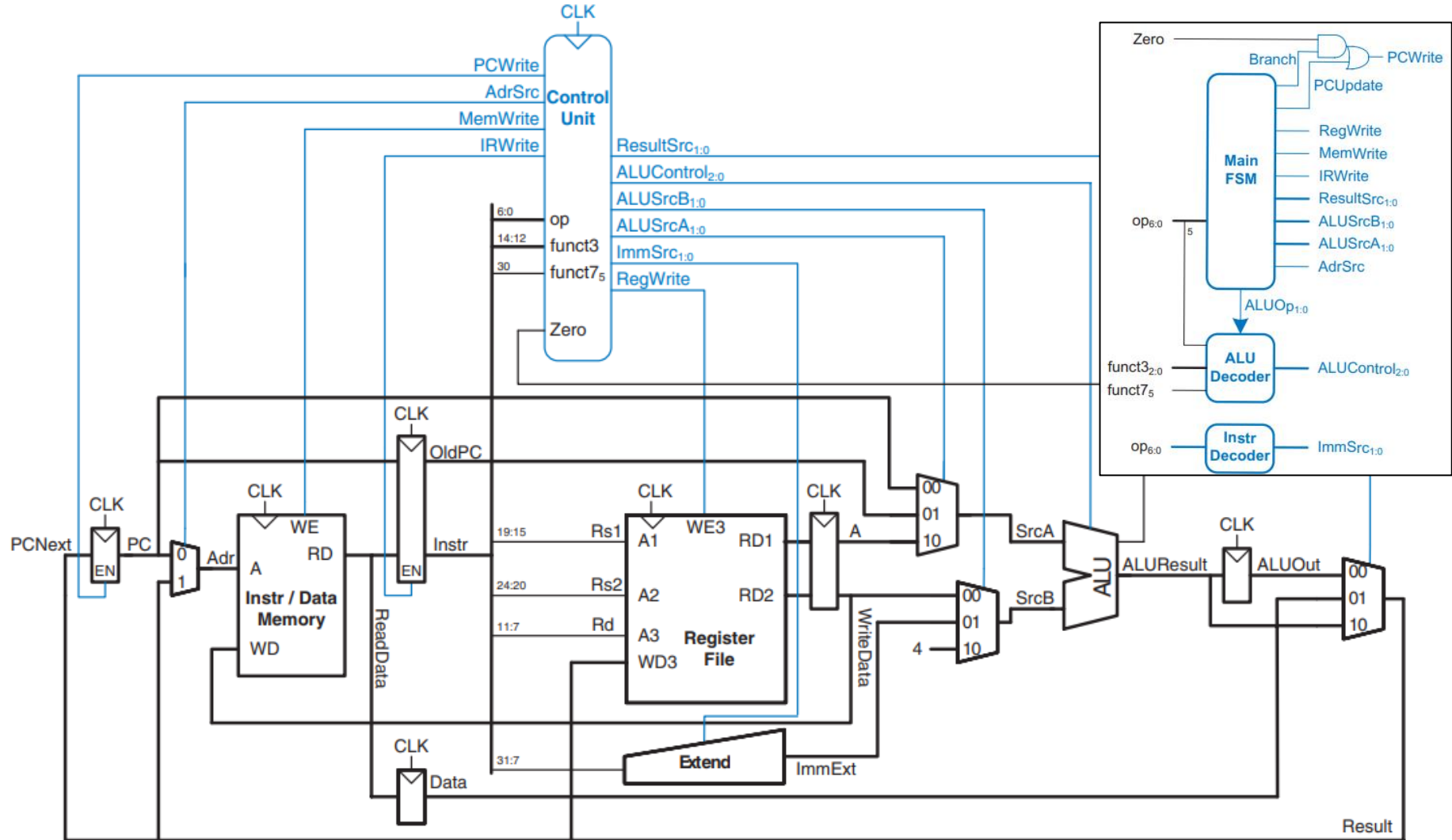
Opcode 01

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
000	nzimm[5]			0			nzimm[4:0]			01		CI c.nop				
000	nzimm[5]			rs1/rd≠0			nzimm[4:0]			01		CI c.addi				
001	imm[11 4 9:8 10 6 7 3:1 5]														01	CJ c.jal
010	imm[5]			rd≠0			imm[4:0]			01		CI c.li				
011	nzimm[9]			2			nzimm[4 6 8:7 5]			01		CI c.addi16sp				
011	nzimm[17]			rd≠{0, 2}			nzimm[16:12]			01		CI c.lui				
100	nzuimm[5]			00	rs1'/rd'			nzuimm[4:0]			01		CI c.srli			
100	nzuimm[5]			01	rs1'/rd'			nzuimm[4:0]			01		CI c.srai			
100	imm[5]			10	rs1'/rd'			imm[4:0]			01		CI c.andi			
100	0			11	rs1'/rd'			00	rs2'			01		CR c.sub		
100	0			11	rs1'/rd'			01	rs2'			01		CR c.xor		
100	0			11	rs1'/rd'			10	rs2'			01		CR c.or		
100	0			11	rs1'/rd'			11	rs2'			01		CR c.and		
101	imm[11 4 9:8 10 6 7 3:1 5]														01	CJ c.j
110	imm[8 4:3]			rs1'			imm[7:6 2:1 5]			01		CB c.beqz				
111	imm[8 4:3]			rs1'			imm[7:6 2:1 5]			01		CB c.bnez				

Opcode 10

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
000			nzuimm[5]				rs1/rd \neq 0				nzuimm[4:0]			10		CI c.slli
000			0				rs1/rd \neq 0				0			10		CI c.slli64
001			uimm[5]				rd				uimm[4:3 8:6]			10		CSS c.fldsp
010			uimm[5]				rd \neq 0				uimm[4:2 7:6]			10		CSS c.lwsp
011			uimm[5]				rd				uimm[4:2 7:6]			10		CSS c.flwsp
100			0				rs1 \neq 0				0			10		CJ c.jr
100			0				rd \neq 0				rs2 \neq 0			10		CR c.mv
100			1				0				0			10		CI c.ebreak
100			1				rs1 \neq 0				0			10		CJ c.jalr
100			1				rs1/rd \neq 0				rs2 \neq 0			10		CR c.add
101							uimm[5:3 8:6]				rs2			10		CSS c.fsdsp
110							uimm[5:2 7:6]				rs2			10		CSS c.swsp
111							uimm[5:2 7:6]				rs2			10		CSS c.fwsp

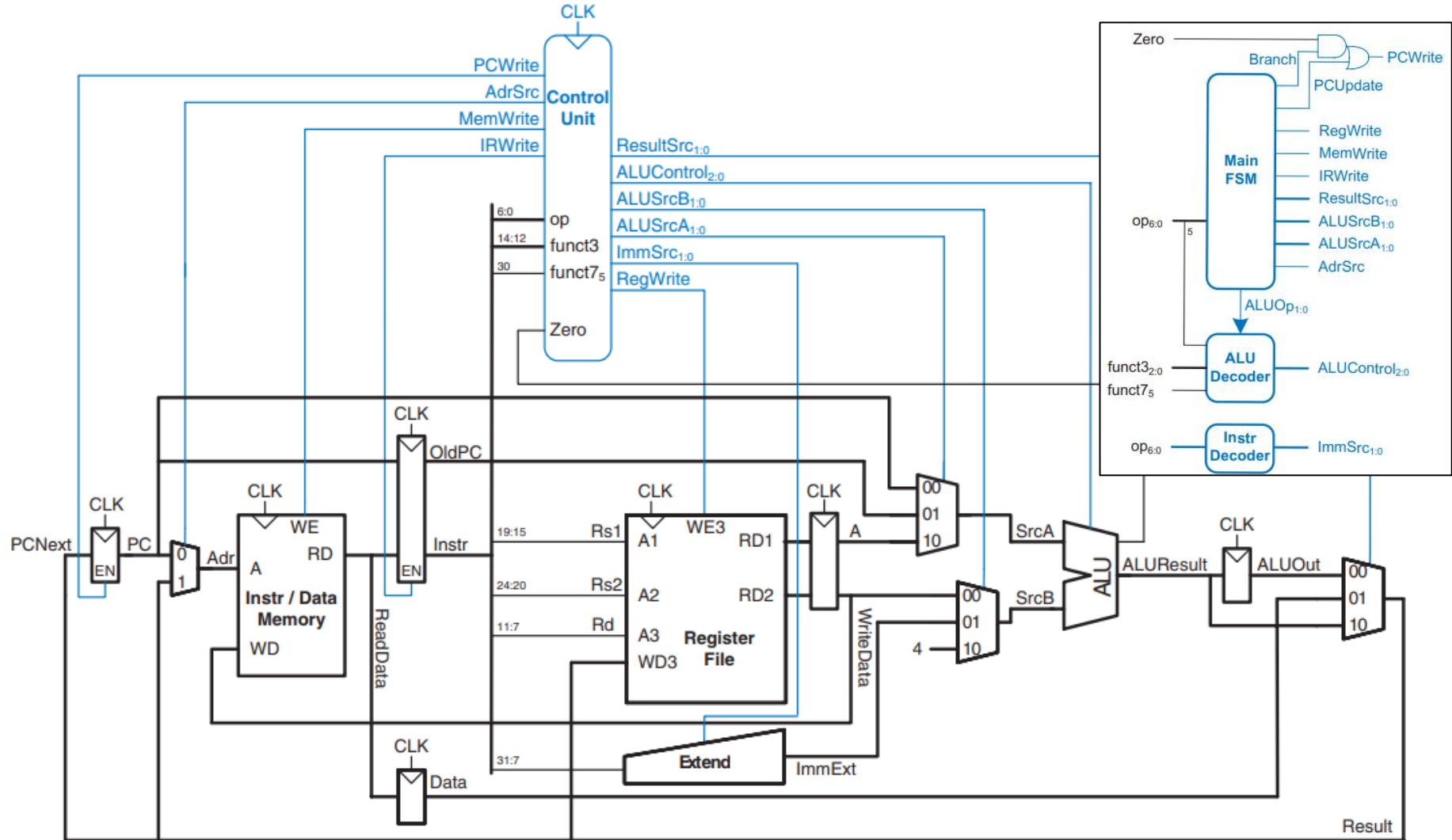


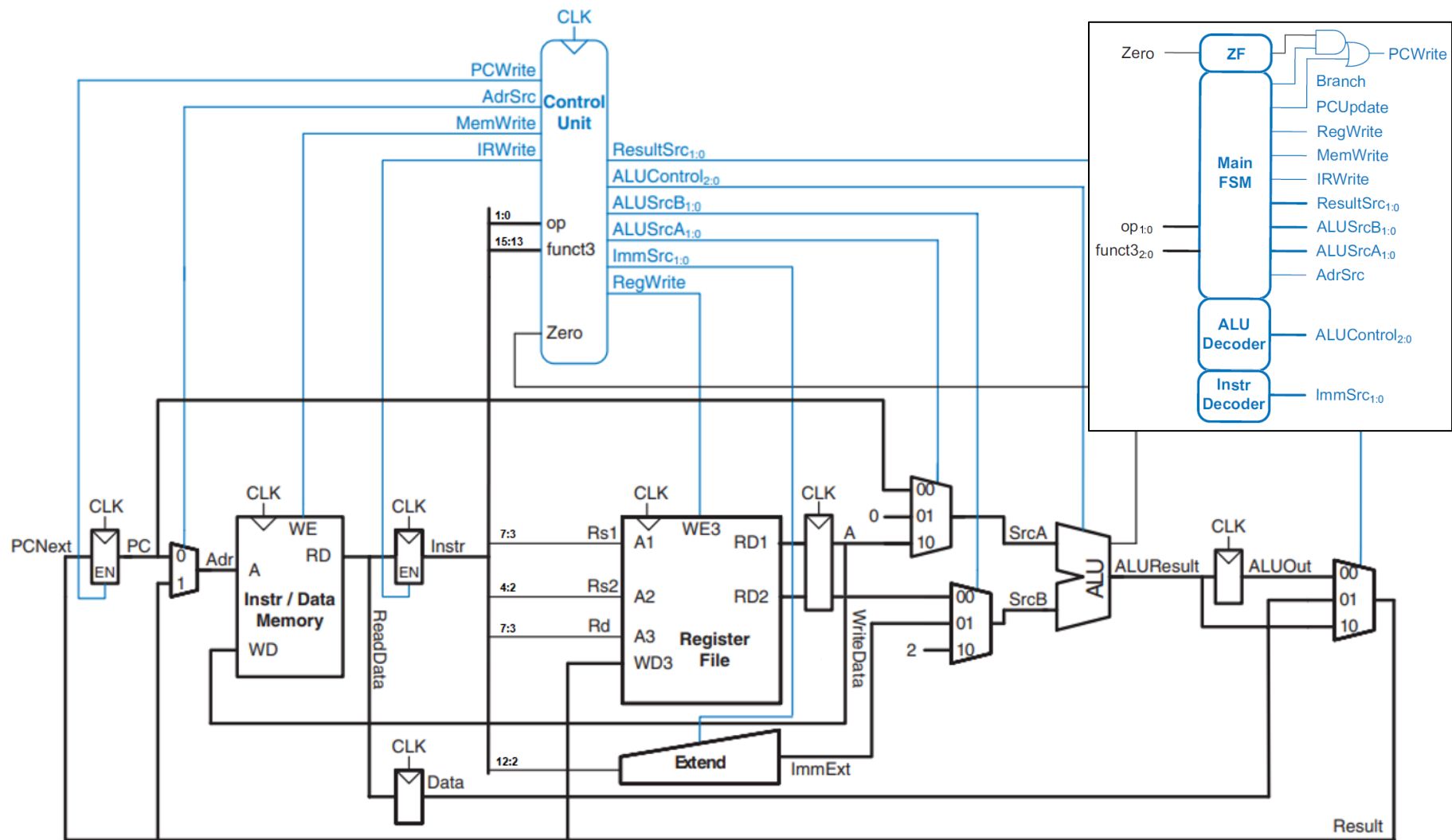


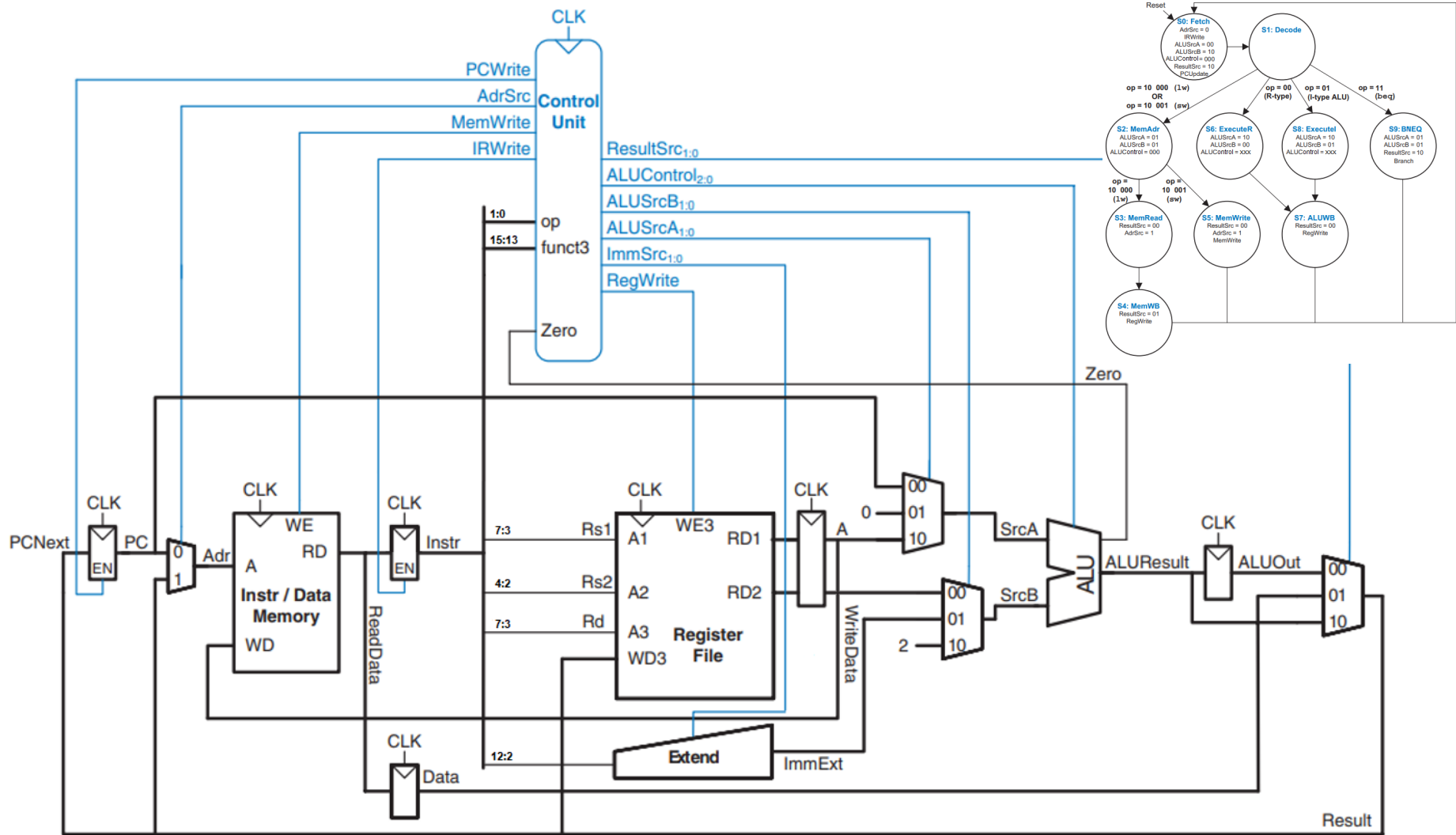
RV16Cm

Formato	15 14 13	12 11 10 9 8	7 6 5	4 3 2	1 0
CRm	Funct3	00000	rd/rs1	rs2	Opcode
CI _m	Funct3	Imm[7:3]	rd/rs1	Imm[2:0]	Opcode
CL _m	Funct3	Addr[7:3]	rd	Addr[2:0]	Opcode
CS _m	Funct3	Addr[7:3]	rs1	Addr[2:0]	Opcode
CB _m	Funct3	Addr[7:3]	Addr[10:8]	Addr[2:0]	Opcode

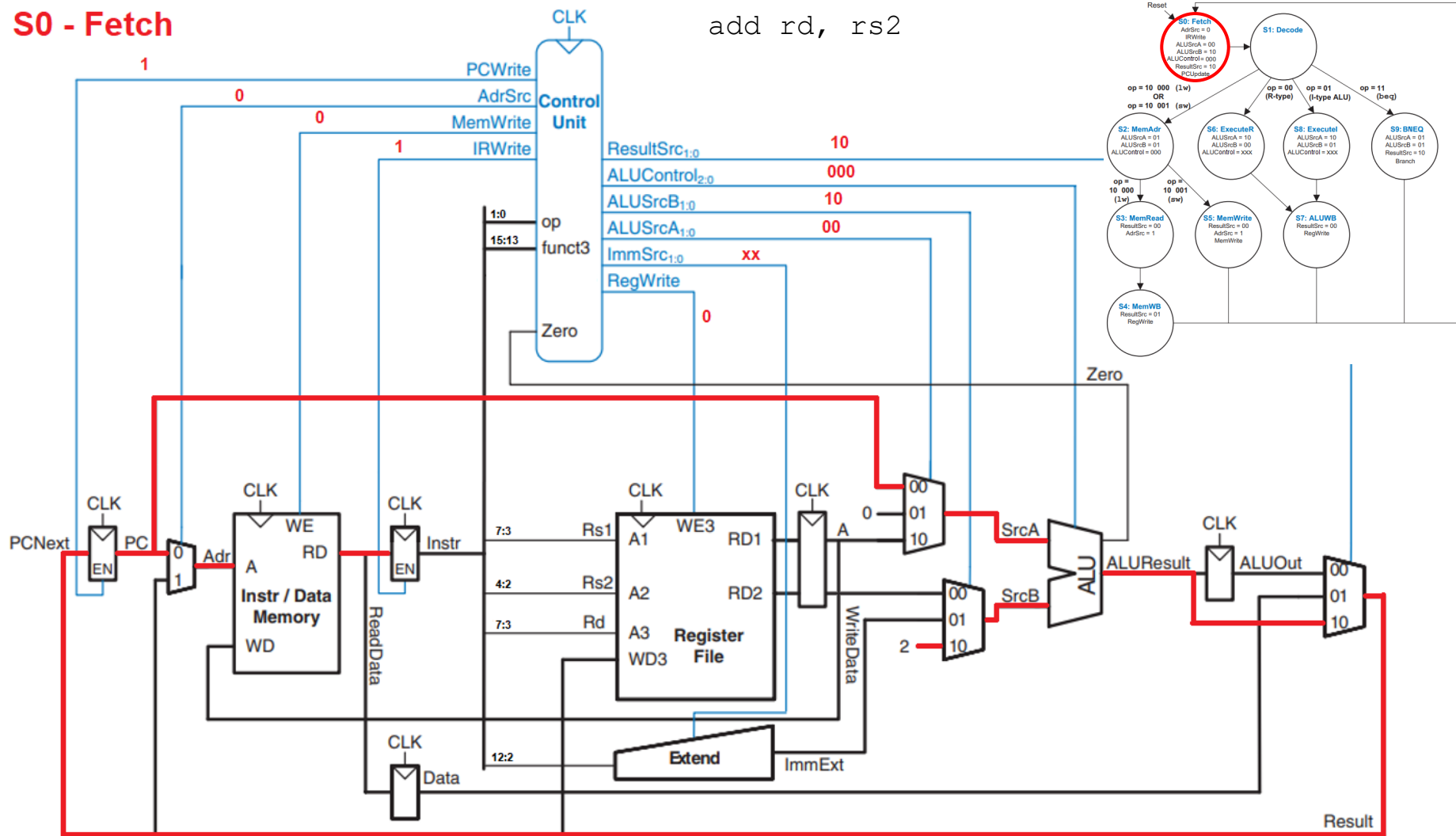
Formato	Instrução	Operação	ALUOp	Funct3		rd/rs1			rs2			Op	
CRm	add rd, rs2	rd <- rd + rs2	.000	0 0 0	0 0 0 0 0	d	d	d	s	s	s	0	0
	sub rd, rs2	rd <- rd - rs2	.001	0 0 1	0 0 0 0 0	d	d	d	s	s	s	0	0
	and rd, rs2	rd <- rd AND rs2	.010	0 1 0	0 0 0 0 0	d	d	d	s	s	s	0	0
	or rd, rs2	rd <- rd OR rs2	.011	0 1 1	0 0 0 0 0	d	d	d	s	s	s	0	0
	slt rd, rs2	rd <- 1 se rd < rs, 0 caso contrário	.101	1 0 1	0 0 0 0 0	d	d	d	s	s	s	0	0
Formato	Instrução	Operação	ALUOp	Funct3	Imm8[7:3]	rd/rs1			Imm8[2:0]			Op	
CIIm	addi rd, Imm8	rd <- rd + s_ext(Imm8)	.000	0 0 0	i i i i i	d	d	d	i	i	i	0	1
Formato	Instrução	Operação	ALUOp	Funct3	Addr8[7:3]	rd			Addr8[2:0]			Op	
CLm	lw rd, Addr8	rd <- [z_ext(Addr8)]	.xxx	0 0 0	a a a a a	d	d	d	a	a	a	1	0
Formato	Instrução	Operação	ALUOp	Funct3	Addr8[7:3]	rs1			Addr8[2:0]			Op	
CSm	sw rs1, Addr8	[z_ext(Addr8)] <- rs1	.xxx	0 0 1	a a a a a	s	s	s	a	a	a	1	0
Formato	Instrução	Operação	ALUOp	Funct3	Addr11[7:3]	Addr11[10:8]			Addr11[2:0]			Op	
CBm	bneqz Addr11	PC <- z_ext(Addr11) se ALUResult ≠ 0	.xxx	0 0 0	a a a a a	a	a	a	a	a	a	1	1





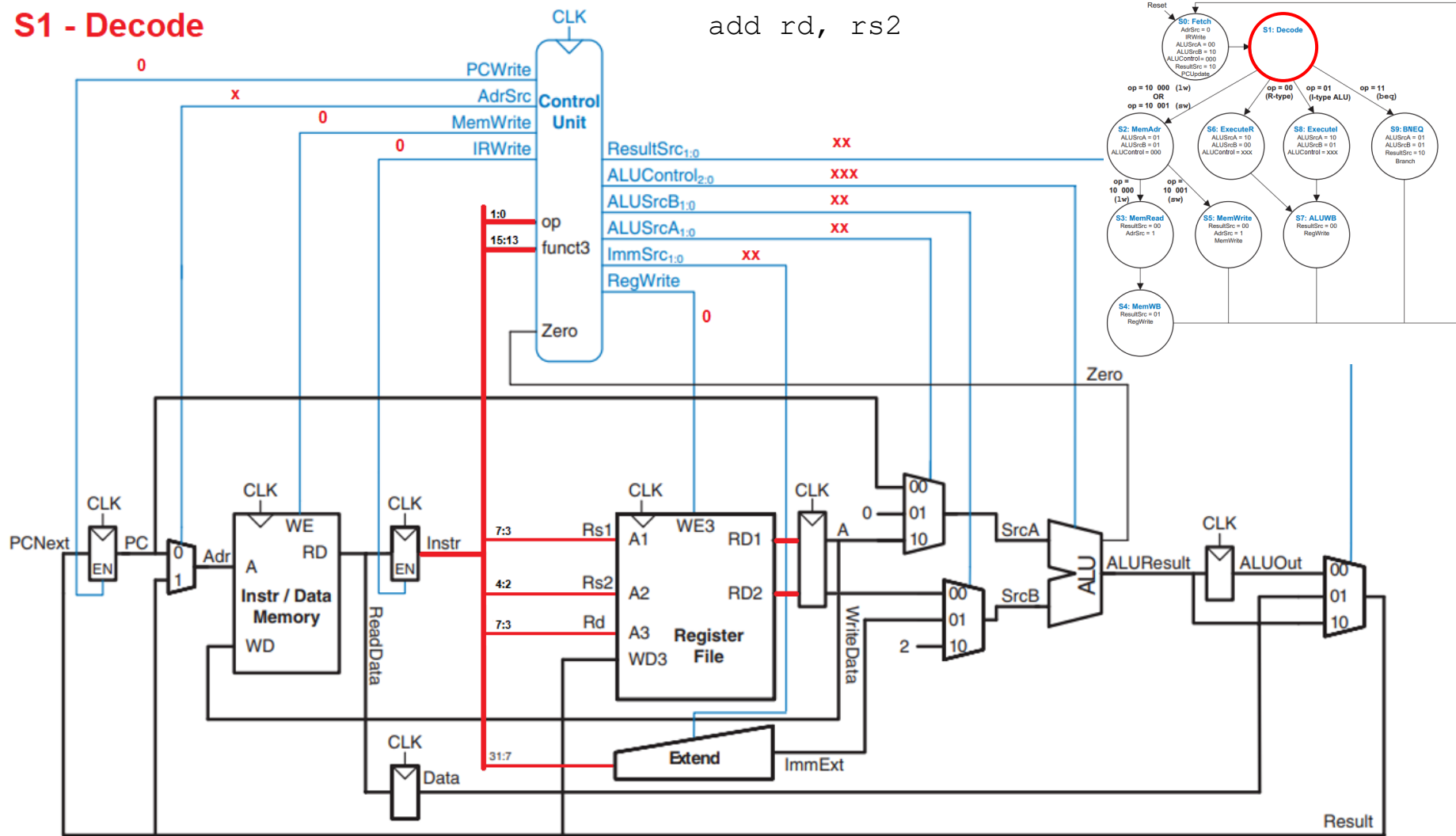



```
add rd, rs2
```



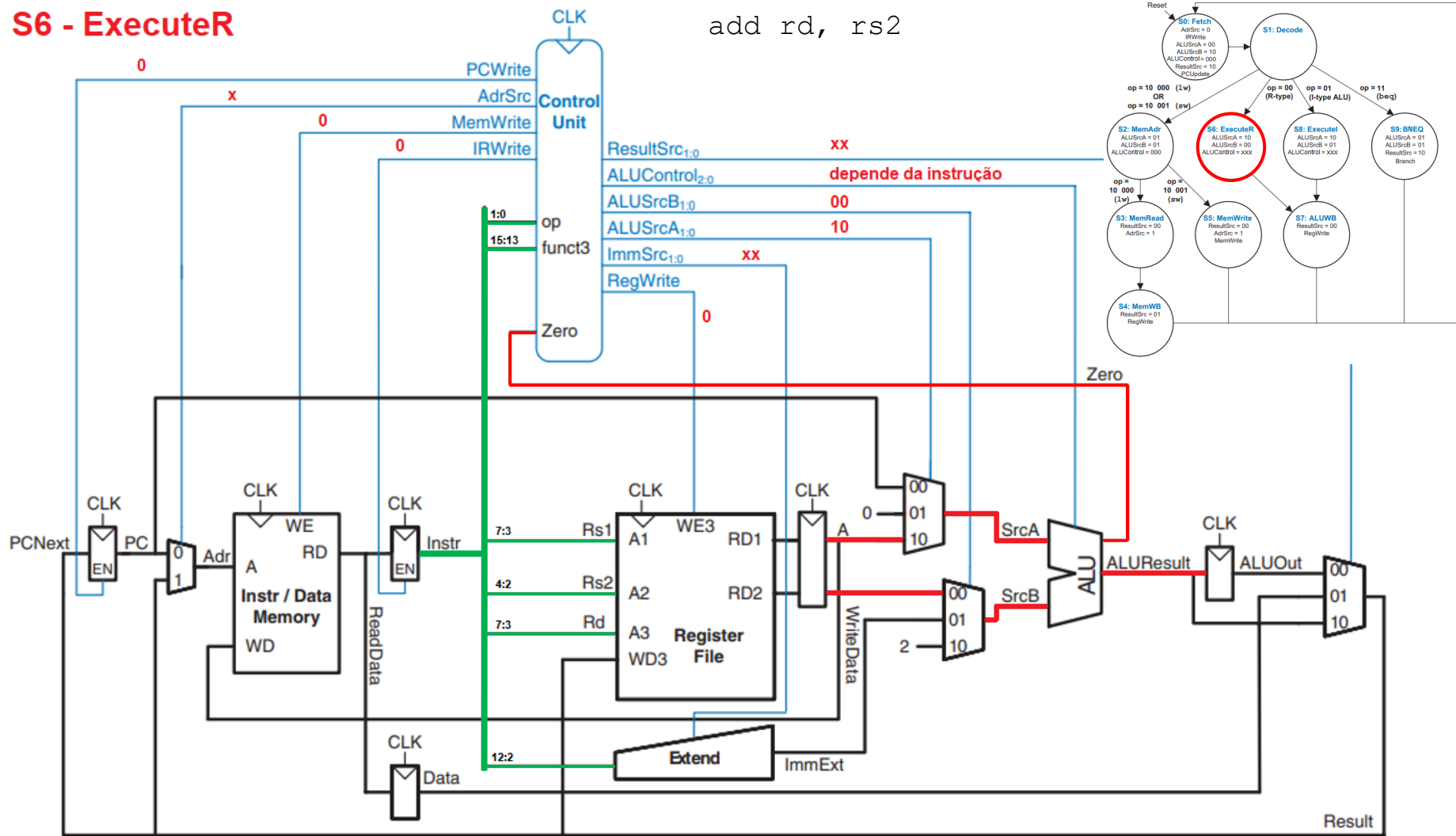
S1 - Decode

add rd, rs2



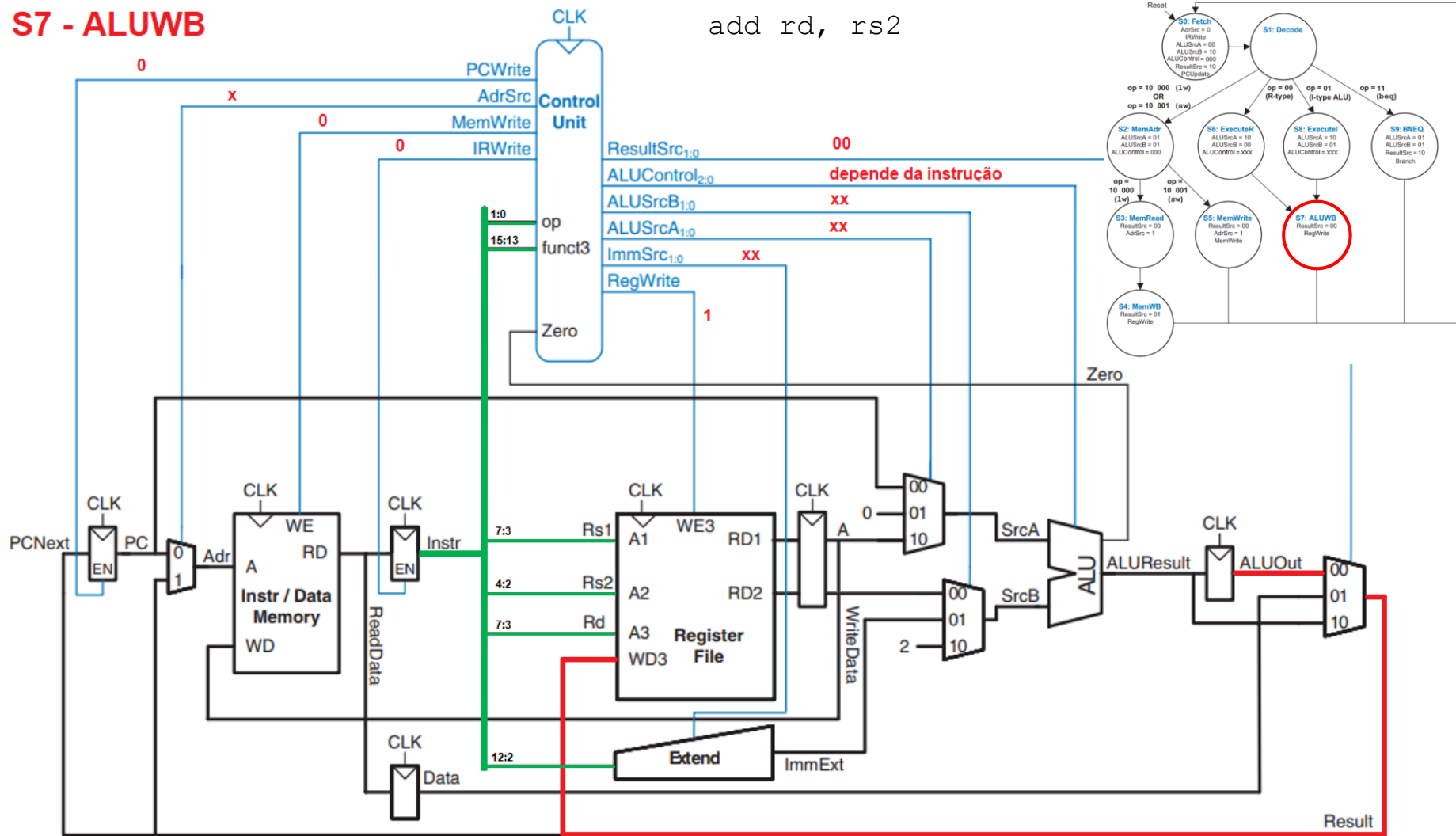
S6 - ExecuteR

add rd, rs2



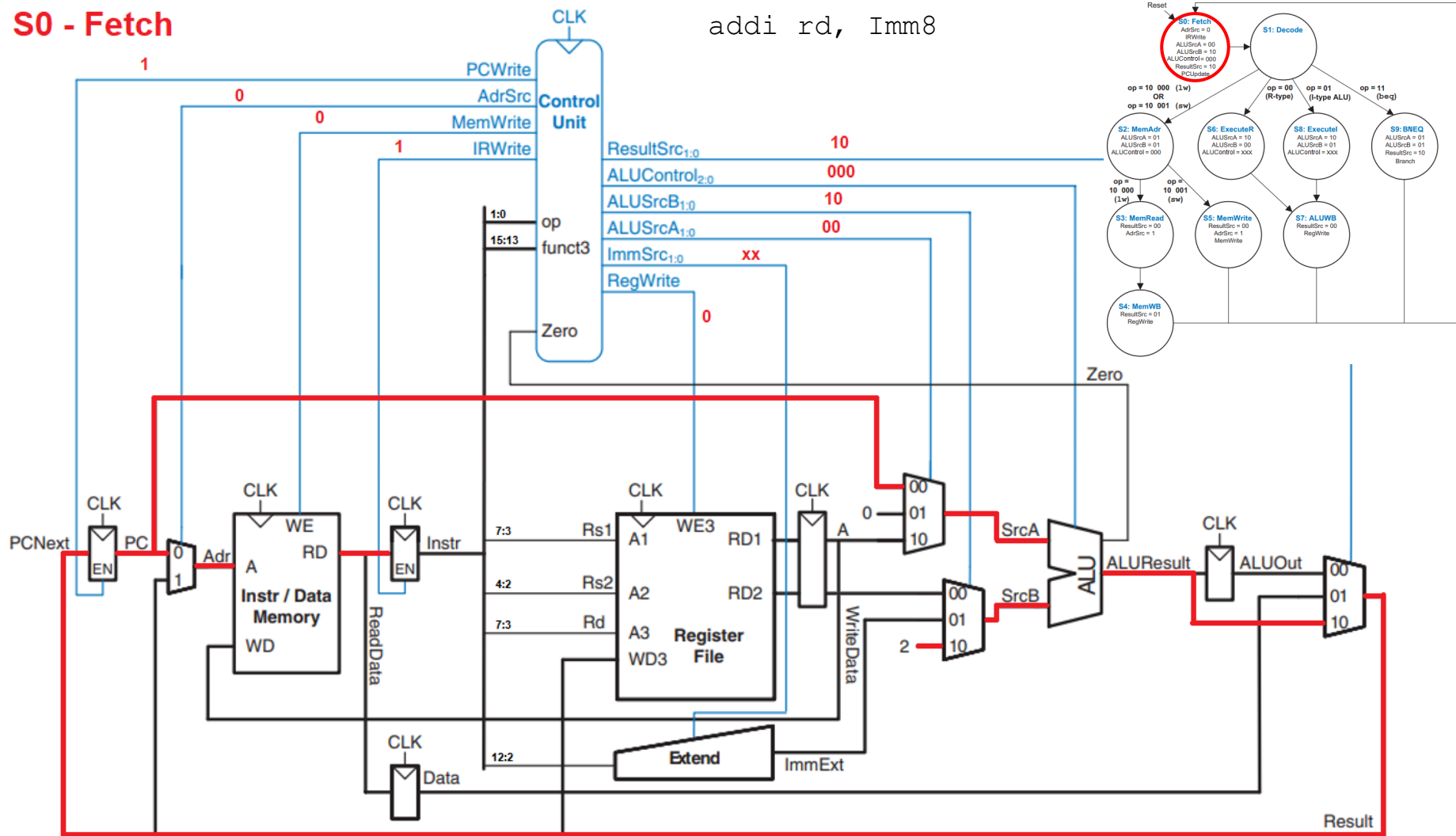
S7 - ALUWB

add rd, rs2



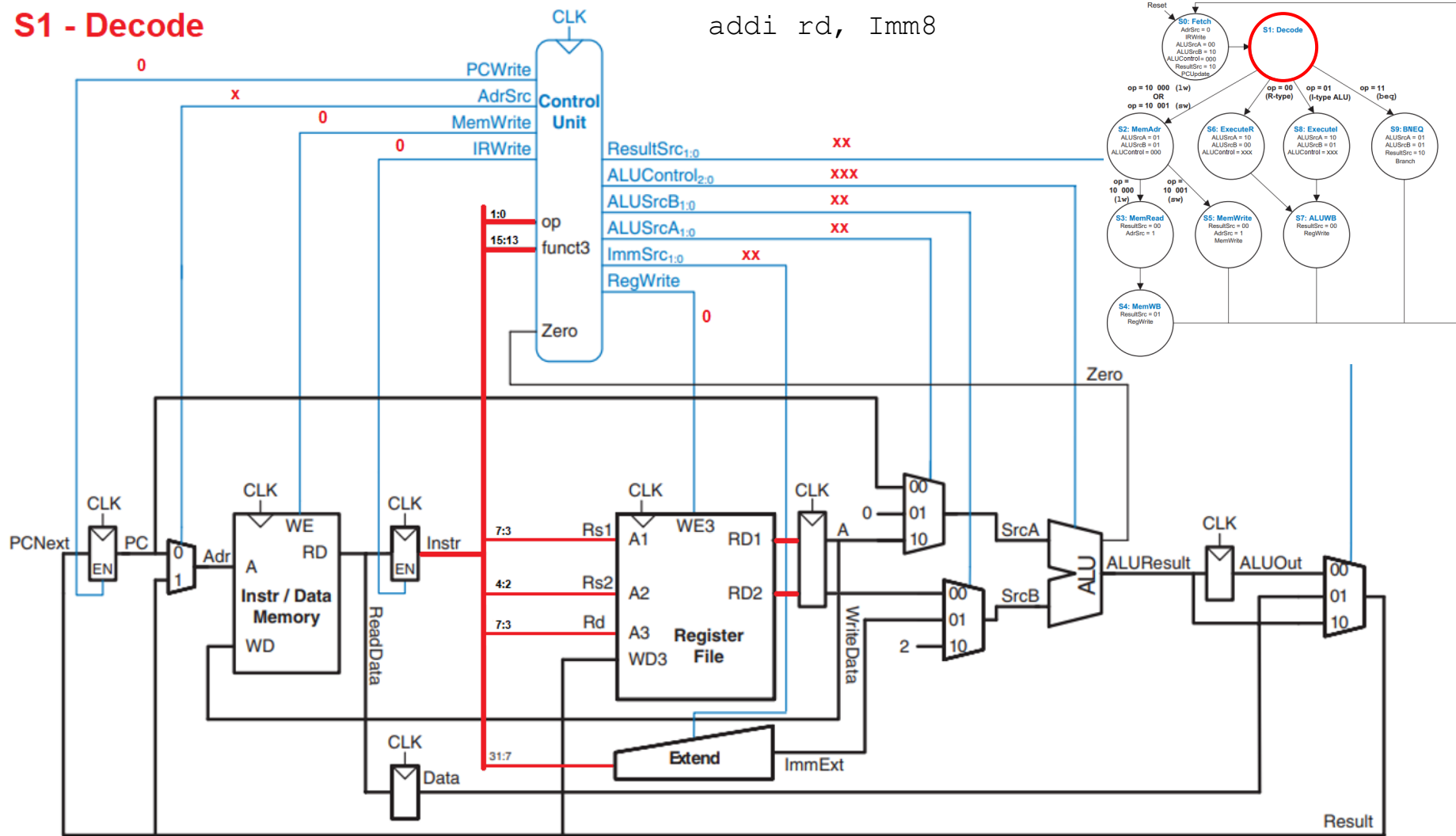
S0 - Fetch

addi rd, Imm8



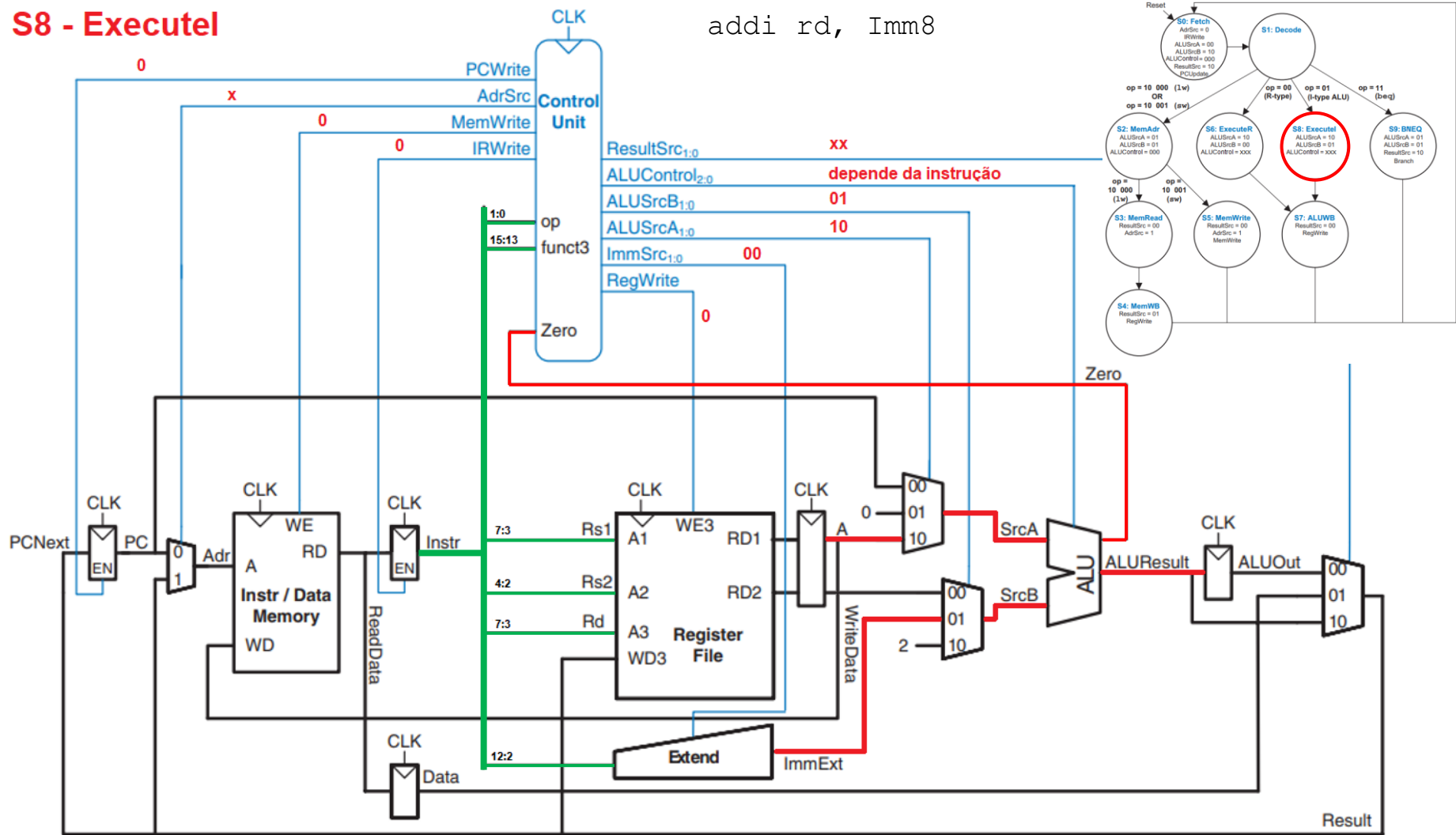
S1 - Decode

addi rd, Imm8



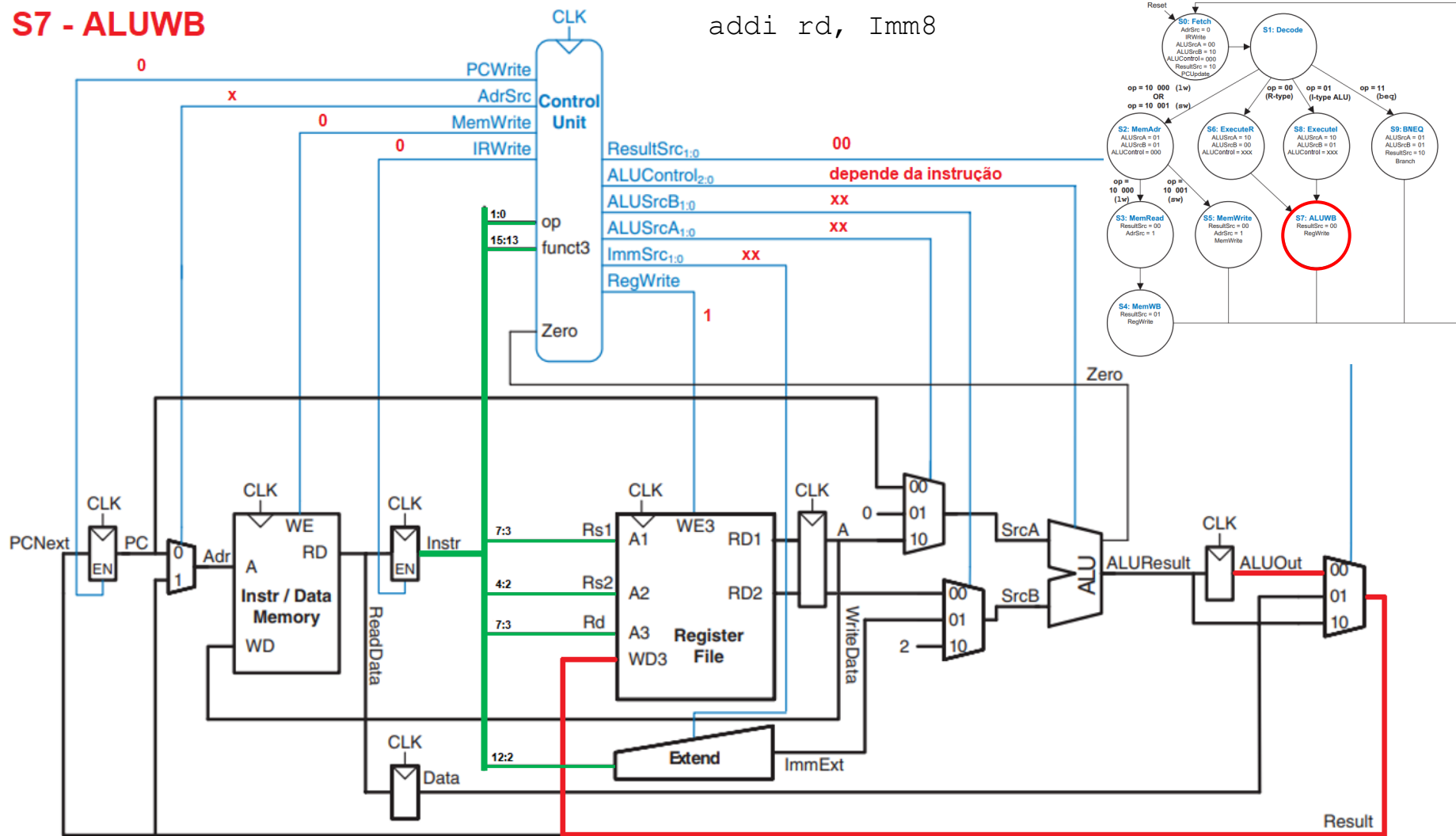
S8 - Executel

addi rd, Imm8



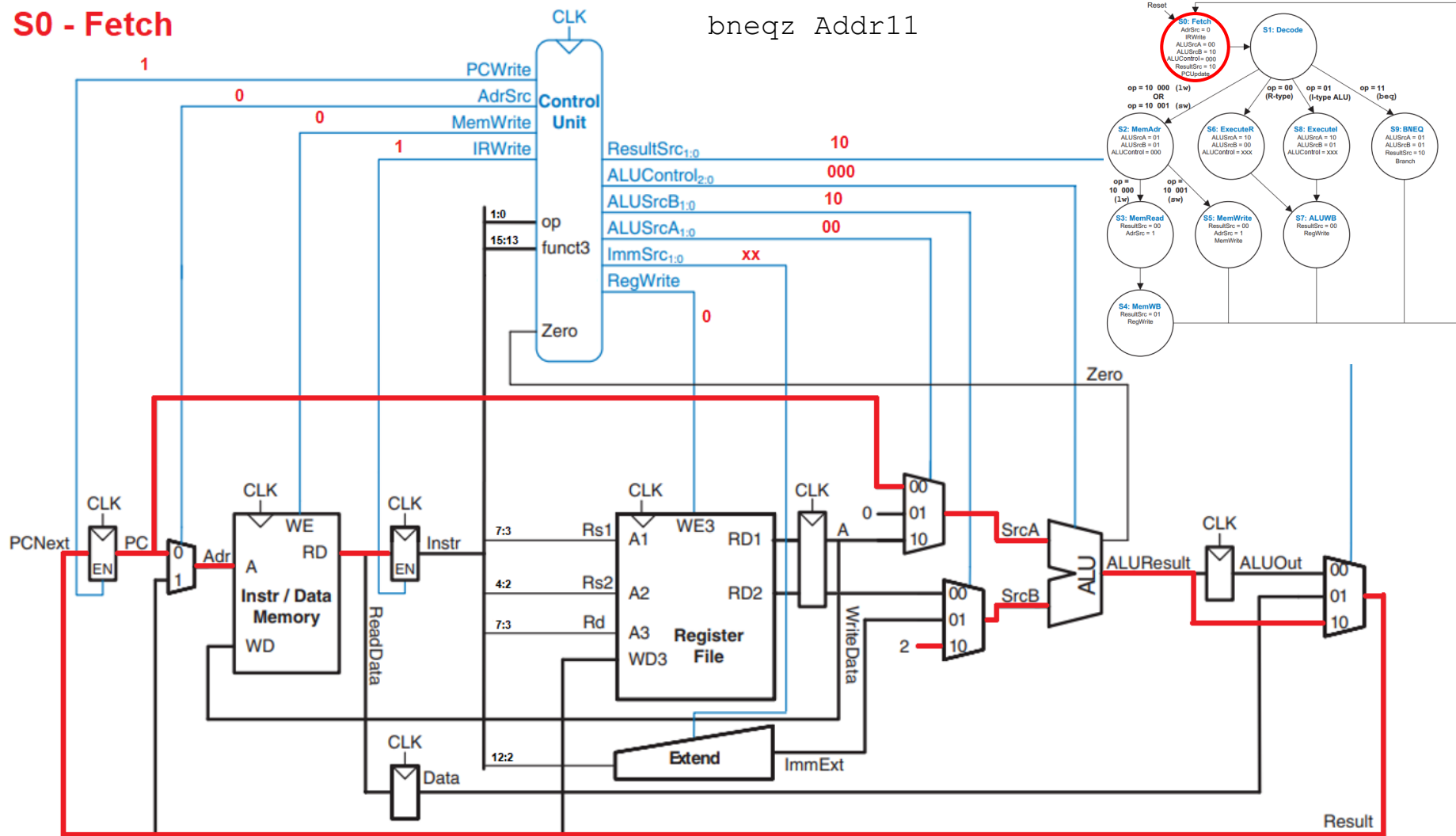
S7 - ALUWB

addi rd, Imm8



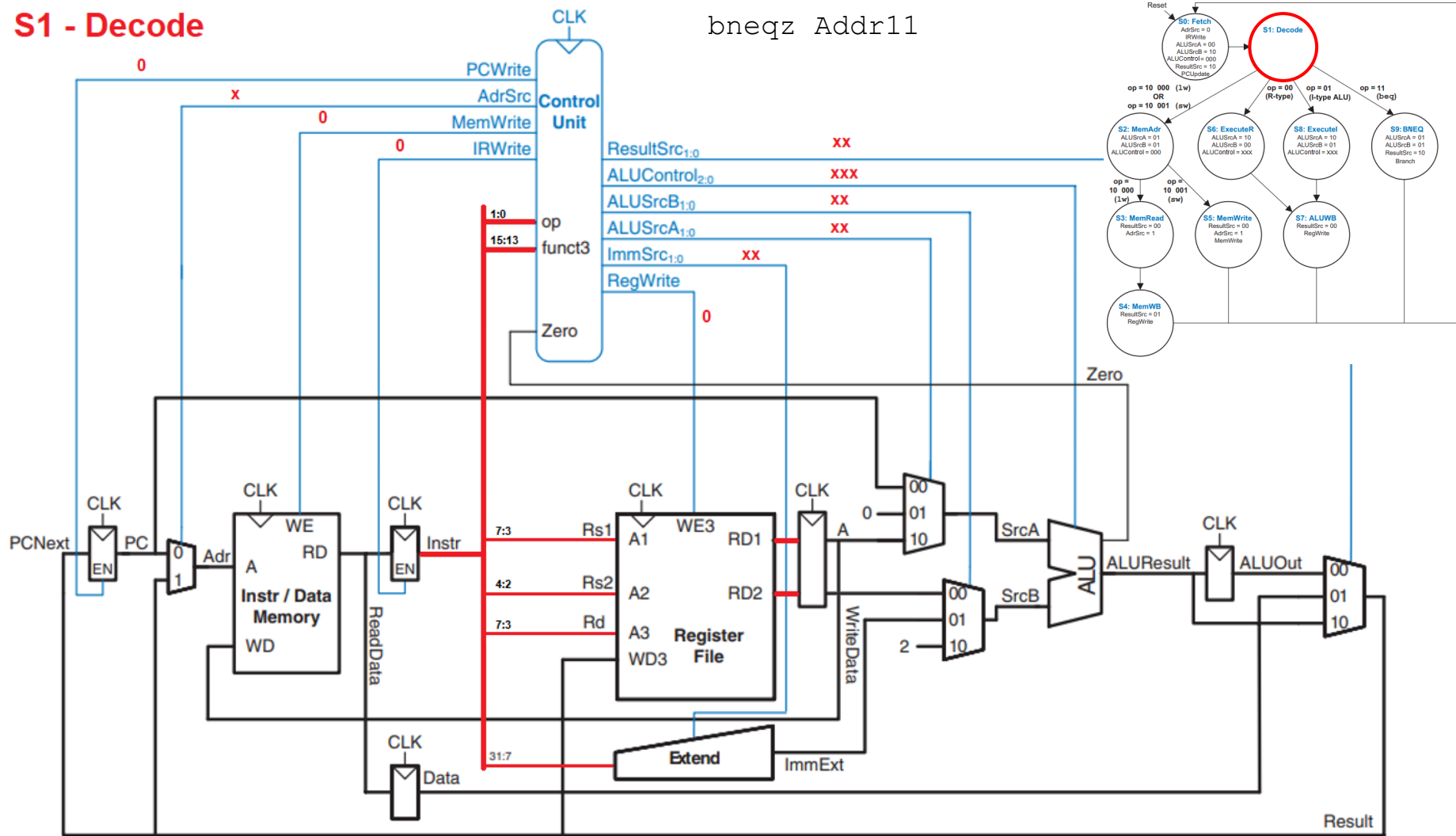
S0 - Fetch

bneqz Addr11



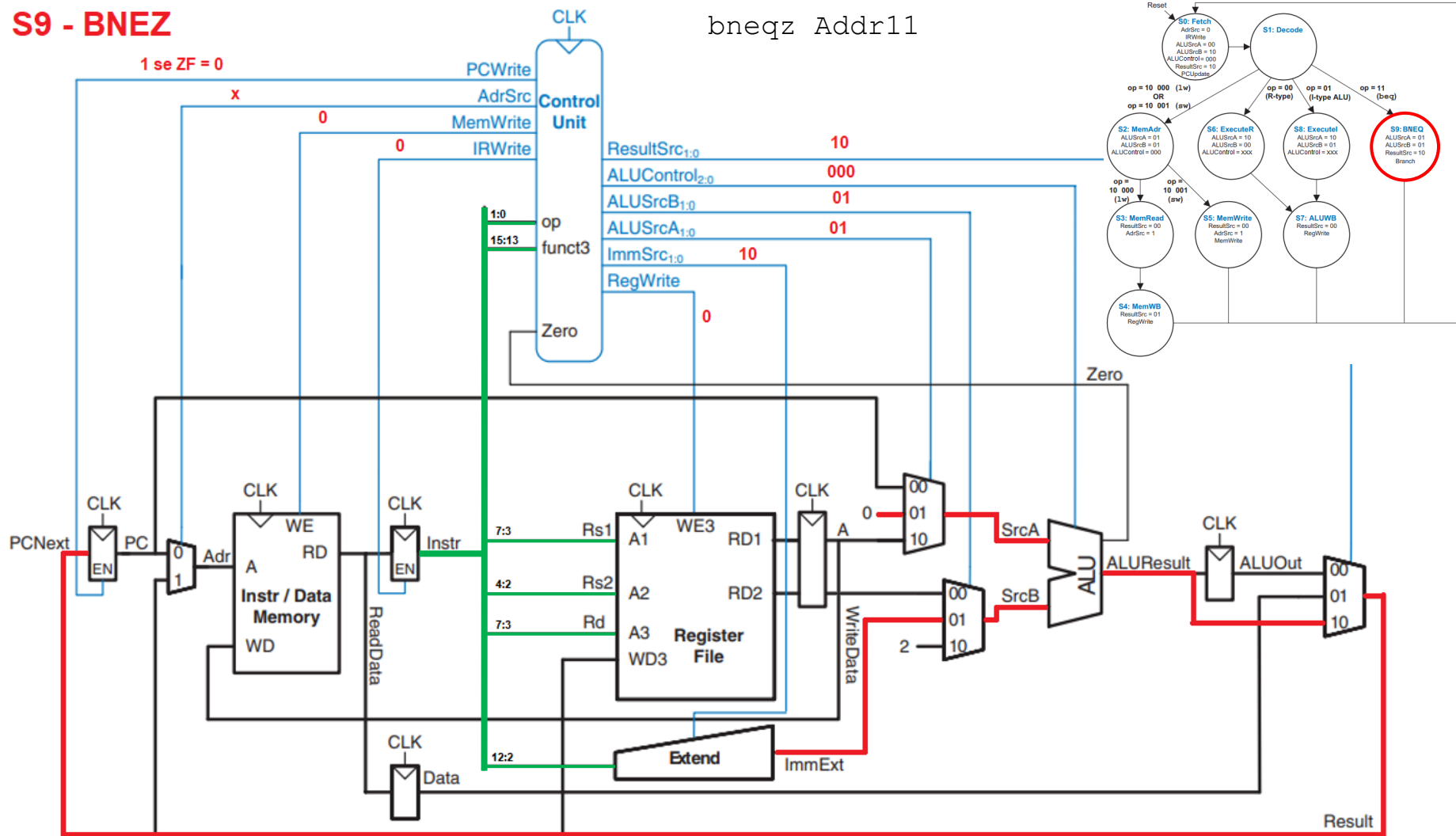
S1 - Decode

bneqz Addr11



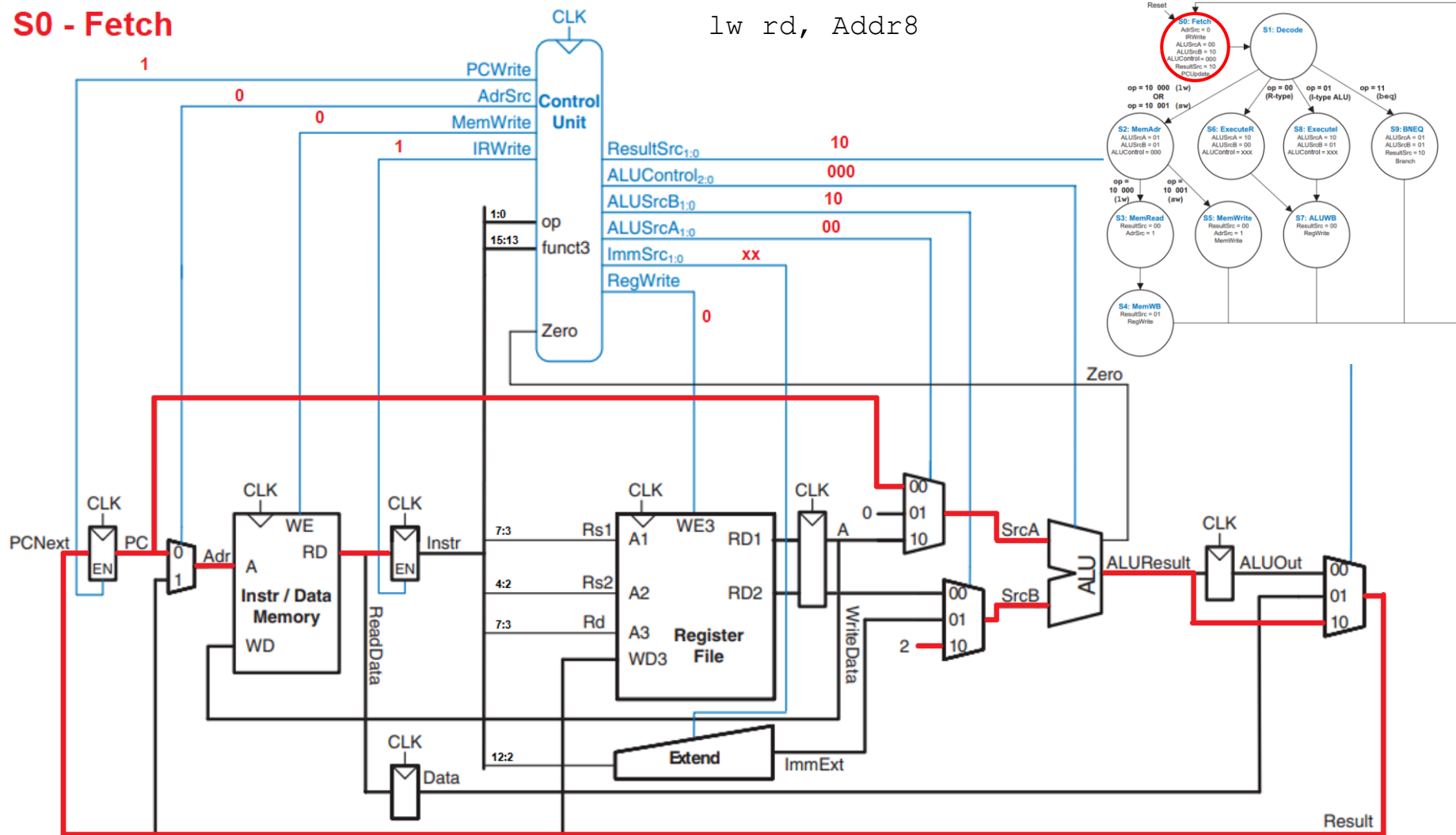
S9 - BNEZ

bneqz Addr11



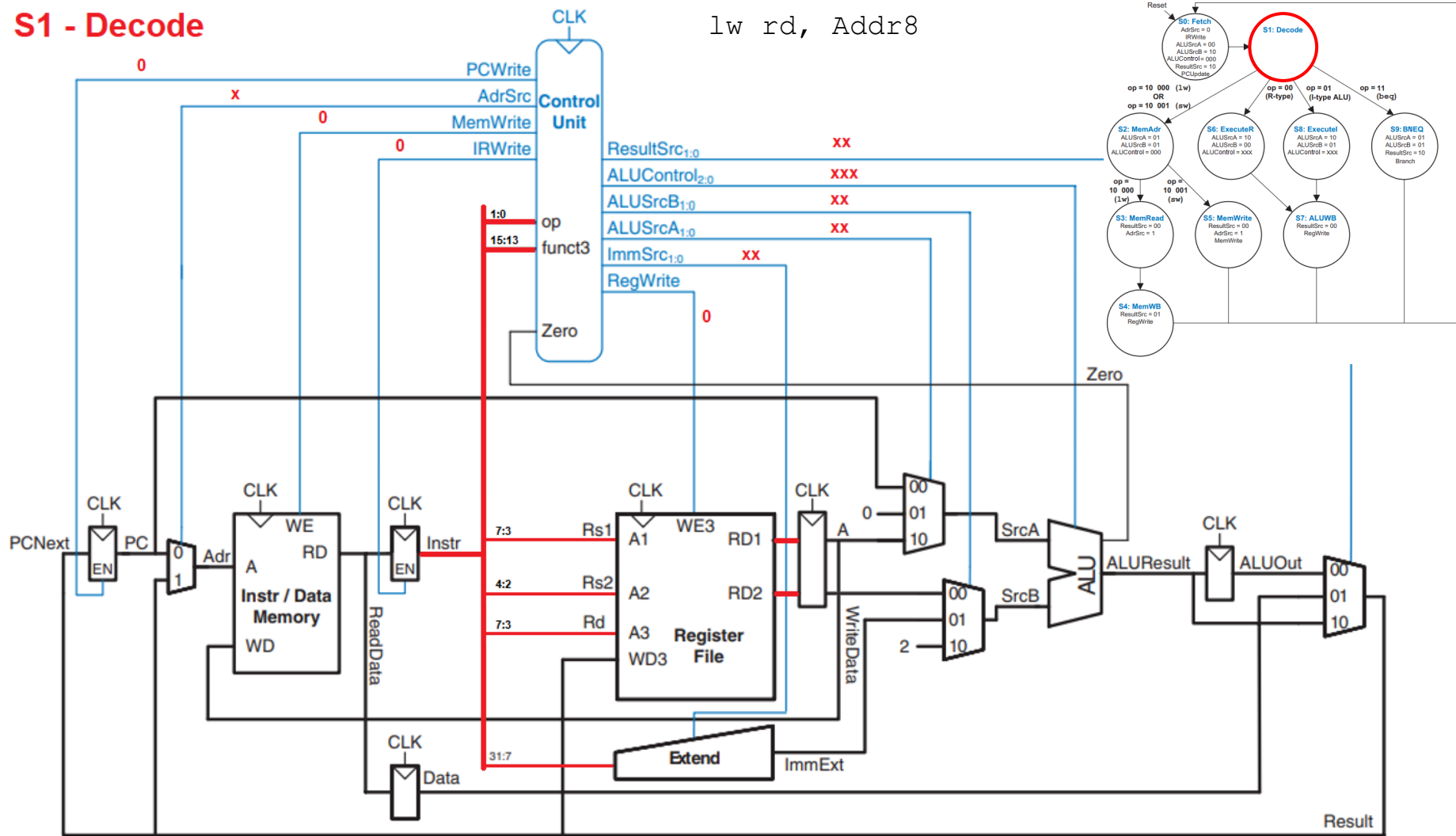
S0 - Fetch

lw rd, Addr8



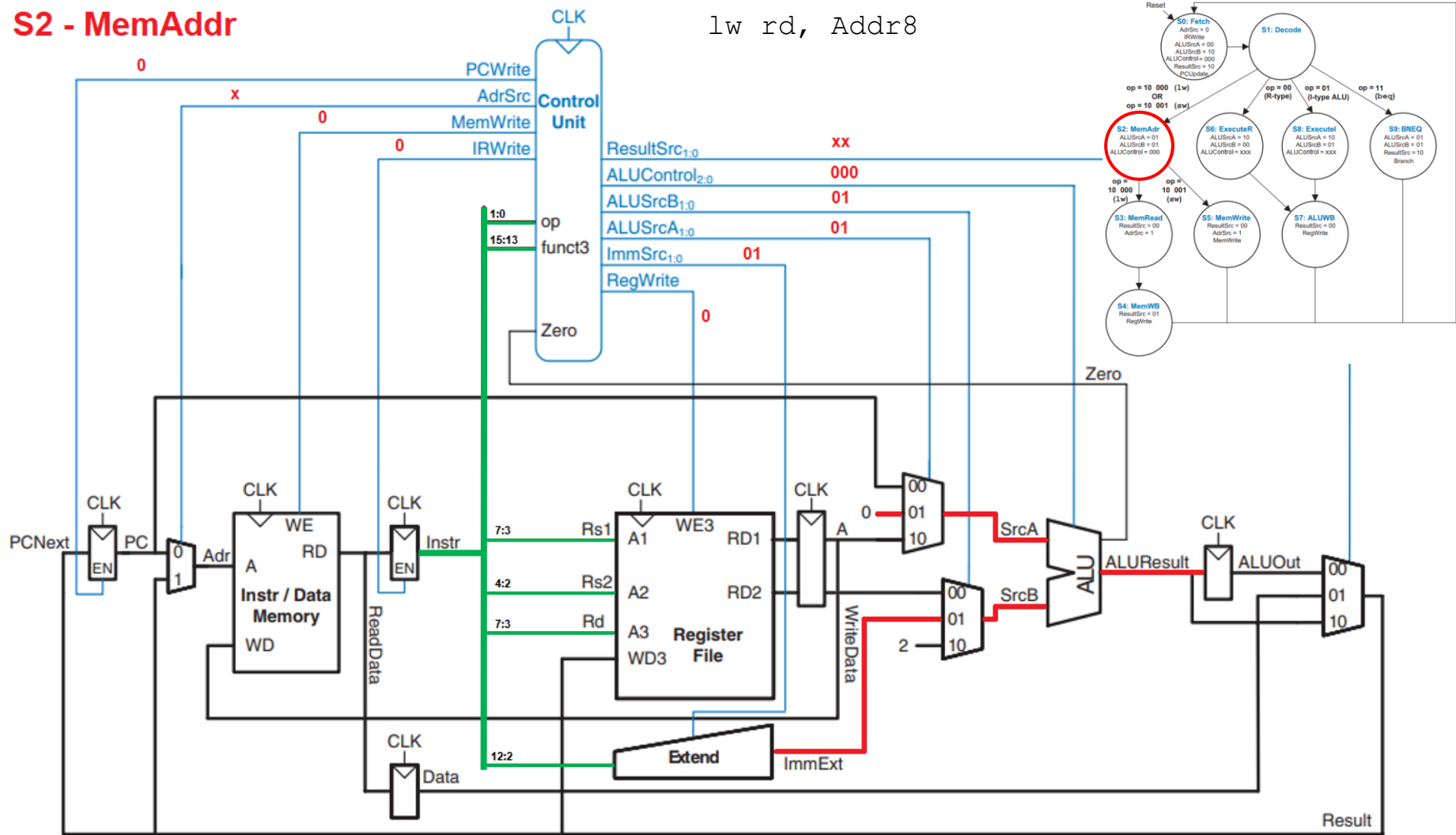
S1 - Decode

lw rd, Addr8



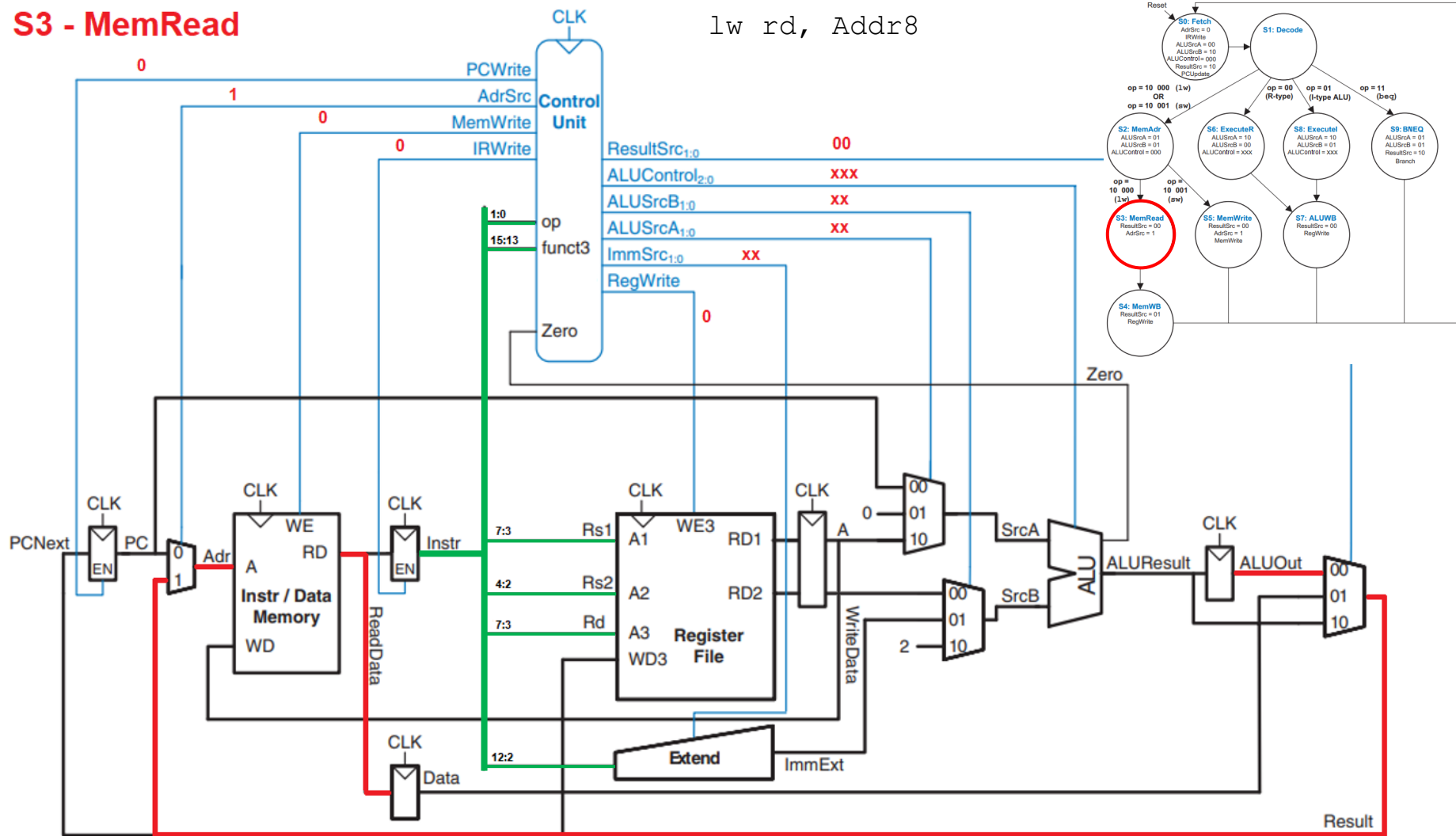
S2 - MemAddr

lw rd, Addr8



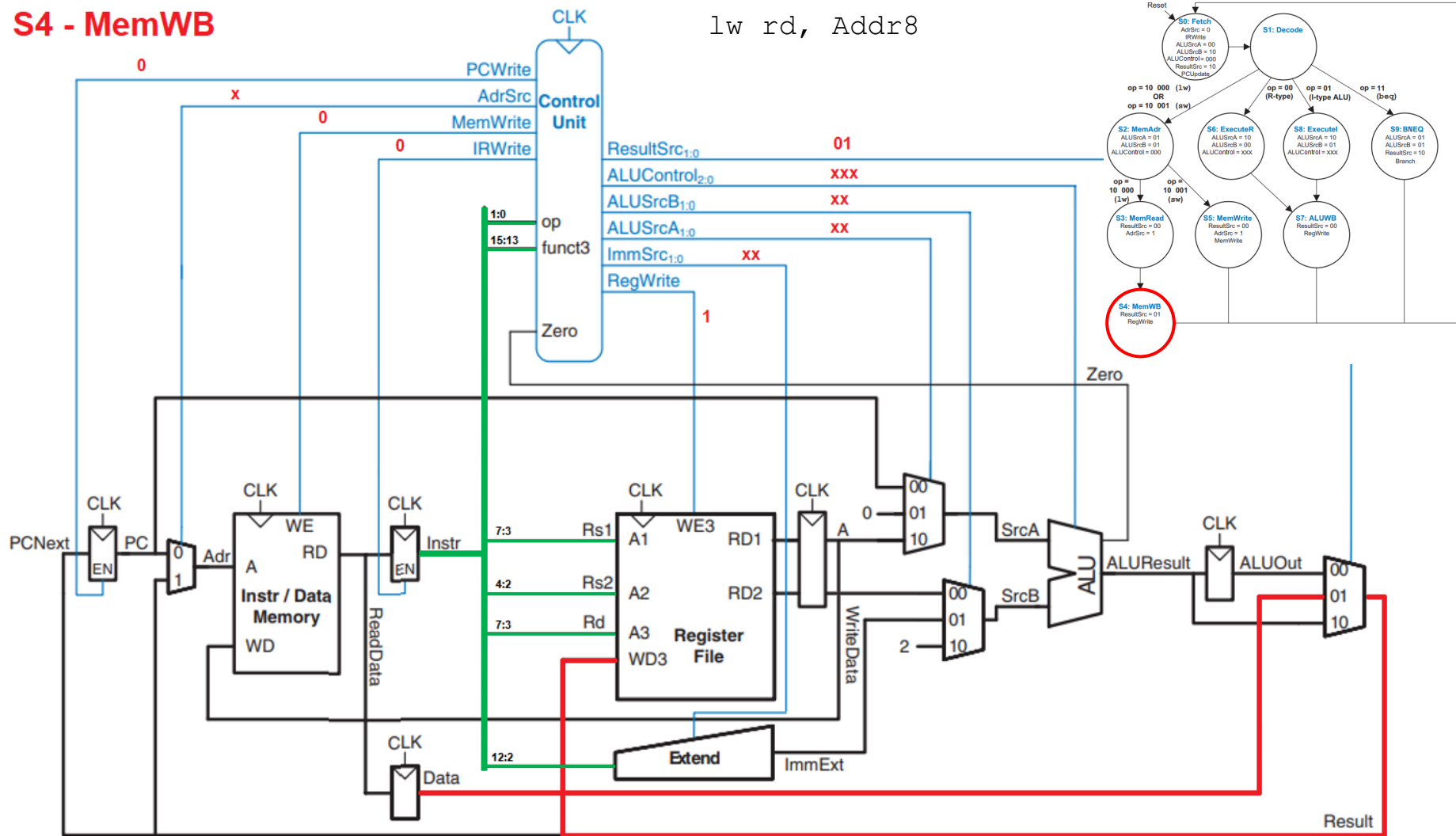
S3 - MemRead

lw rd, Addr8



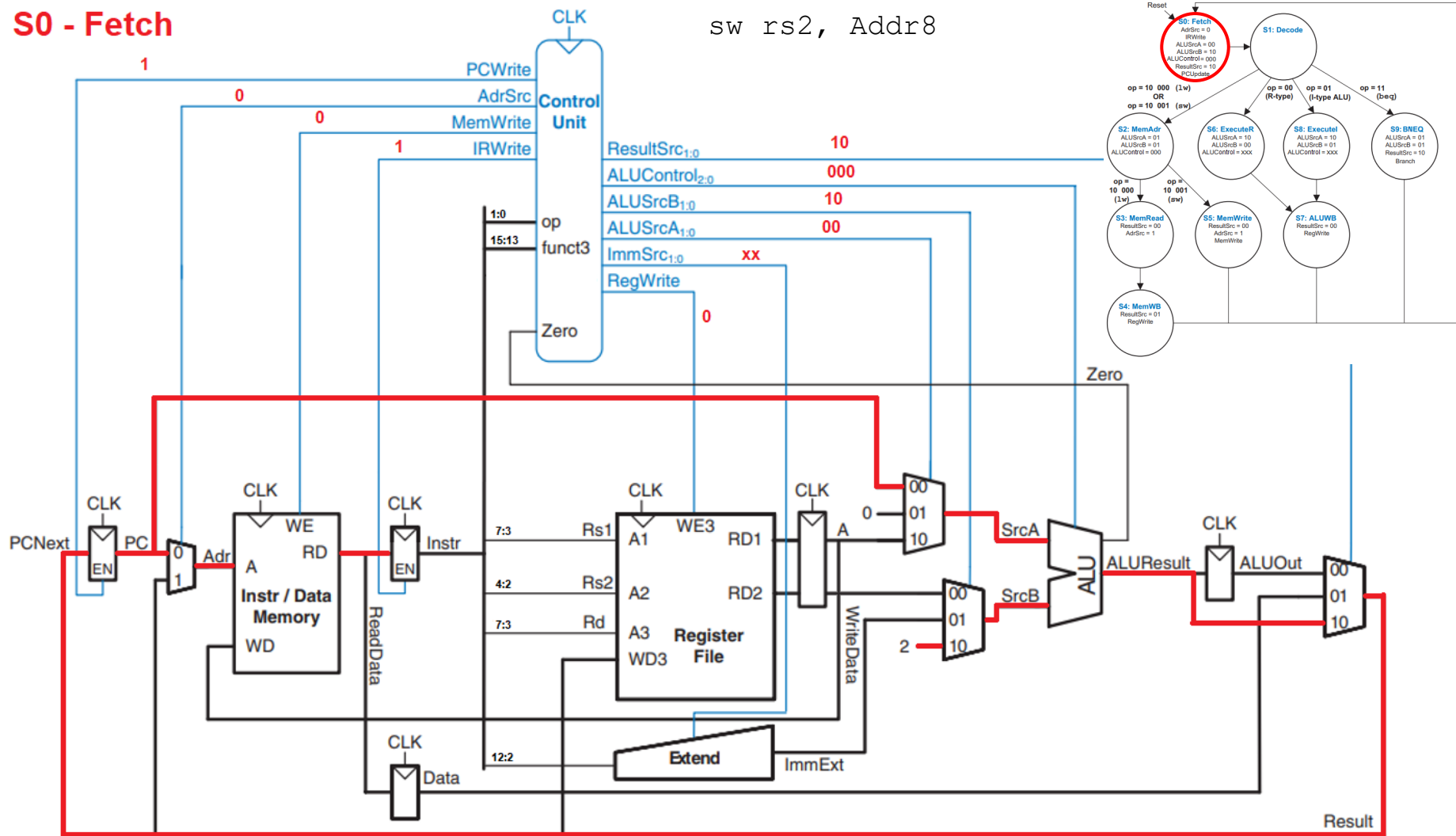
S4 - MemWB

lw rd, Addr8



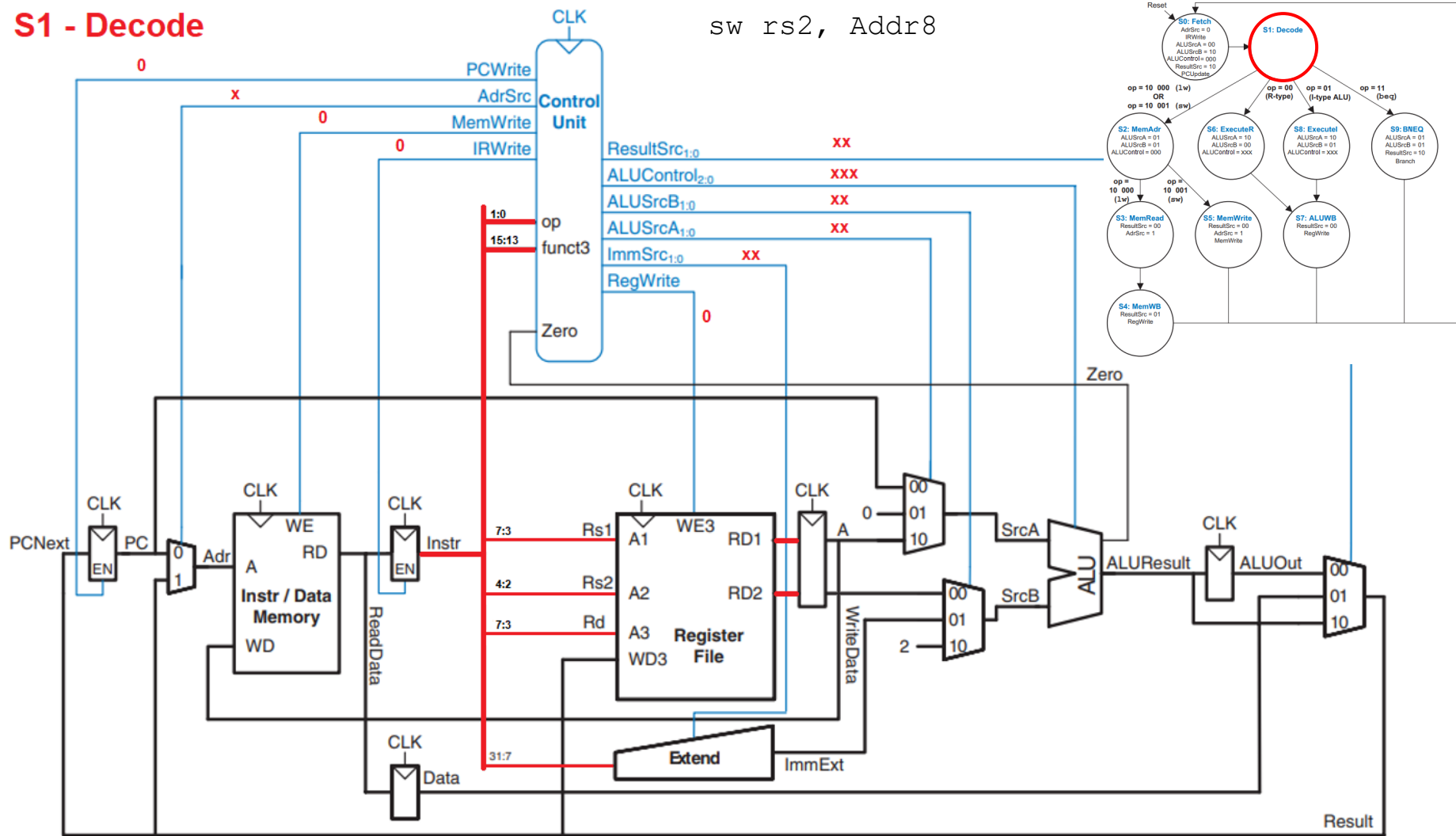
S0 - Fetch

sw rs2, Addr8



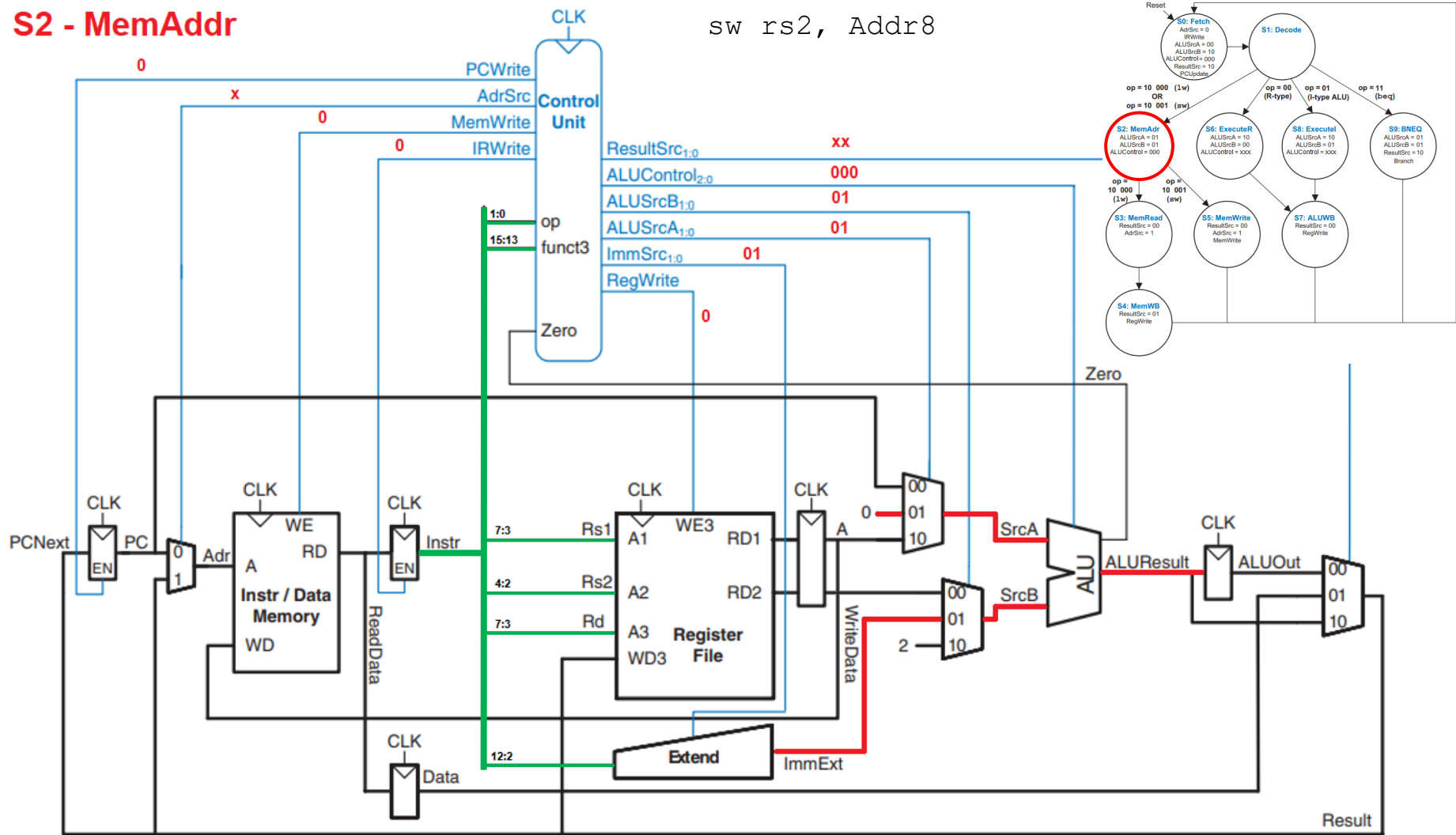
S1 - Decode

sw rs2, Addr8



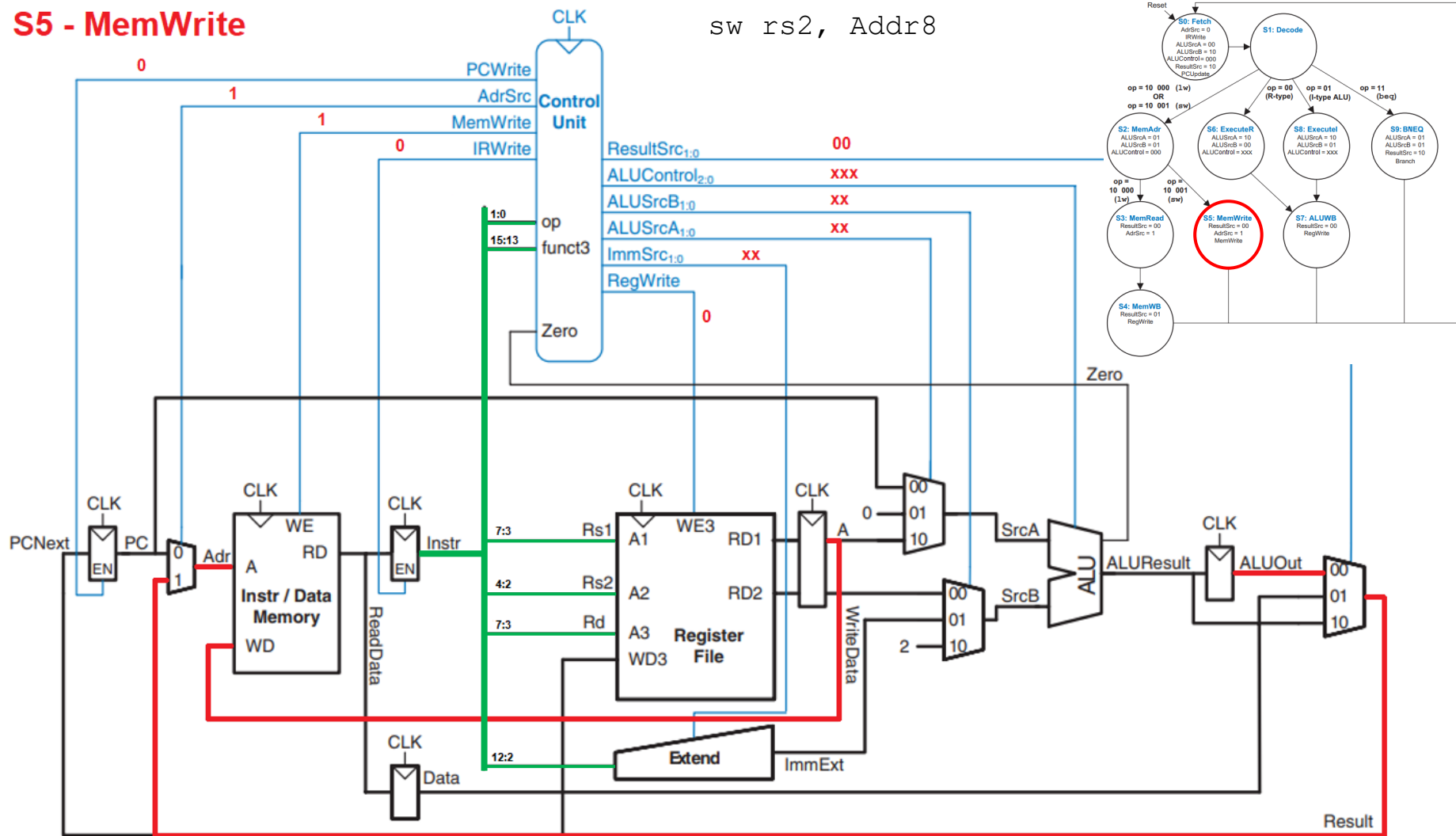
S2 - MemAddr

sw rs2, Addr8



S5 - MemWrite

sw rs2, Addr8



S0 - Fetch

continua...

