3.18 74/21 = 3 remainder 9

Step	Action	Quotient	Divisor	Remainder
0	Initial Vals	000 000	010 001 000 000	000 000 111 100
1	Rem=Rem–Div	000 000	010 001 000 000	101 111 111 100
	Rem<0, R+D, Q<<	000 000	010 001 000 000	000 000 111 100
	Rshift Div	000 000	001 000 100 000	000 000 111 100
2	Rem=Rem–Div	000 000	001 000 100 000	111 000 011 100
	Rem<0, R+D, Q<<	000 000	001 000 100 000	000 000 111 100
	Rshift Div	000 000	000 100 010 000	000 000 111 100
3	Rem=Rem–Div	000 000	000 100 010 000	111 100 101 100
	Rem<0, R+D, Q<<	000 000	000 100 010 000	000 000 111 100
	Rshift Div	000 000	000 010 001 000	000 000 111 100
4	Rem=Rem–Div	000 000	000 010 001 000	111 110 110 100
	Rem<0, R+D, Q<<	000 000	000 010 001 000	000 000 111 100
	Rshift Div	000 000	000 001 000 100	000 000 111 100
5	Rem=Rem–Div	000 000	000 001 000 100	111 111 111 000
	Rem<0, R+D, Q<<	000 000	000 001 000 100	000 000 111 100
	Rshift Div	000 000	000 000 100 010	000 000 111 100
6	Rem=Rem–Div	000 000	000 000 100 010	000 000 011 010
	Rem>0, Q<<1	000 001	000 000 100 010	000 000 011 010
	Rshift Div	000 001	000 000 010 001	000 000 011 010
7	Rem=Rem–Div	000 001	000 000 010 001	000 000 001 001
	Rem>0, Q<<1	000 011	000 000 010 001	000 000 001 001
	Rshift Div	000 011	000 000 001 000	000 000 001 001

4.1

4.1.1 The values of the signals are as follows:

RegWrite	MemRead	ALUMux	MemWrite	ALUop	RegMux	Branch
0	0	1 (Imm)	1	ADD	X	0

ALUMux is the control signal that controls the Mux at the ALU input, 0 (Reg) selects the output of the register file, and 1 (Imm) selects the immediate from the instruction word as the second input to the ALU.

RegMux is the control signal that controls the Mux at the Data input to the register file, 0 (ALU) selects the output of the ALU, and 1 (Mem) selects the output of memory.

A value of X is a "don't care" (does not matter if signal is 0 or 1)

4.1.2 All except branch Add unit and write port of the Registers

4.1.3 Outputs that are not used: Branch Add, write port of Registers

No outputs: None (all units produce outputs)

4.2

4.2.1 This instruction uses instruction memory, both register read ports, the ALU to add Rd and Rs together, data memory, and write port in Registers.

4.2.2 None. This instruction can be implemented using existing blocks.

4.2.3 None. This instruction can be implemented without adding new control signals. It only requires changes in the Control logic.