Problem 4:

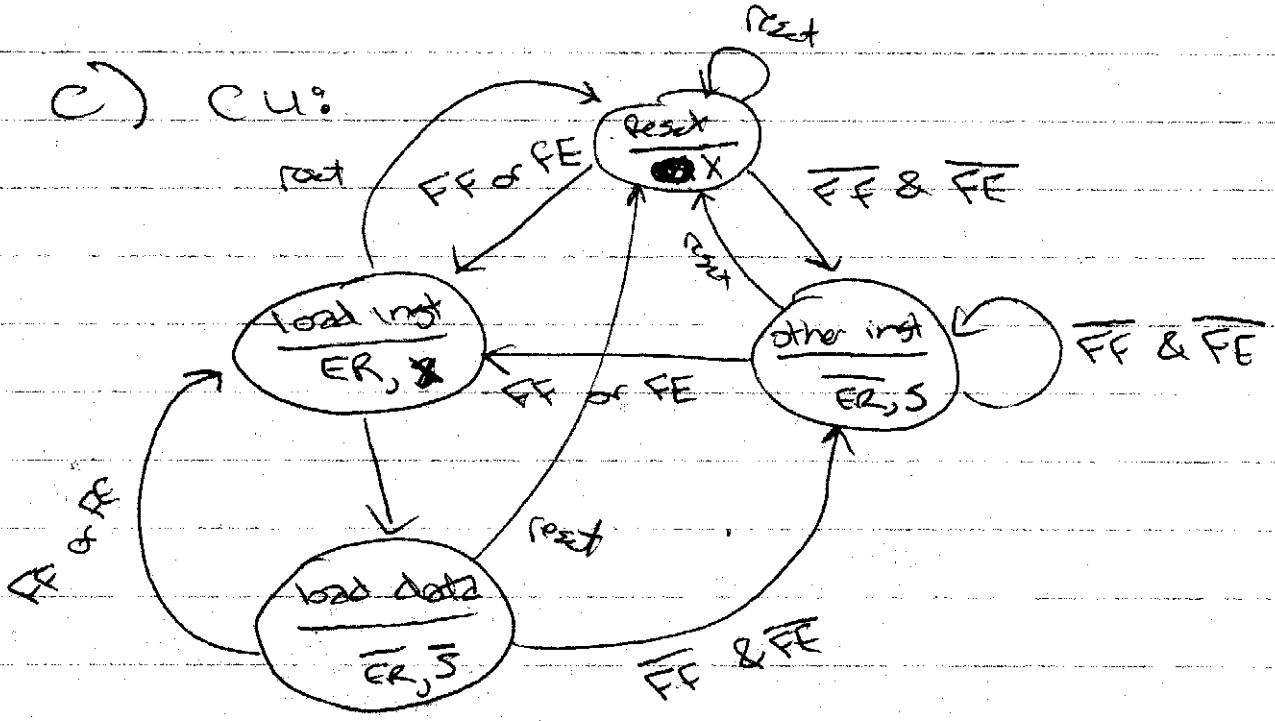
A) R FF FF FE FF FD FF FC FB F7 F9

Inst	Reg A	Reg B	description
FF	x	x	load next byte into A
FF	FF	x	loading this byte into A
FE	FF	x	load next byte into B
FF	FF	FF	loading this byte into B
FD	00	FF	clear A
FF	00	FF	load next byte into A
FC	FC	FF	load this byte into A
FB	FC+FF = FB	FF	add A+B & store in A
F7	04	FF	invert A
F9	04	FF	AND A&B & store in A

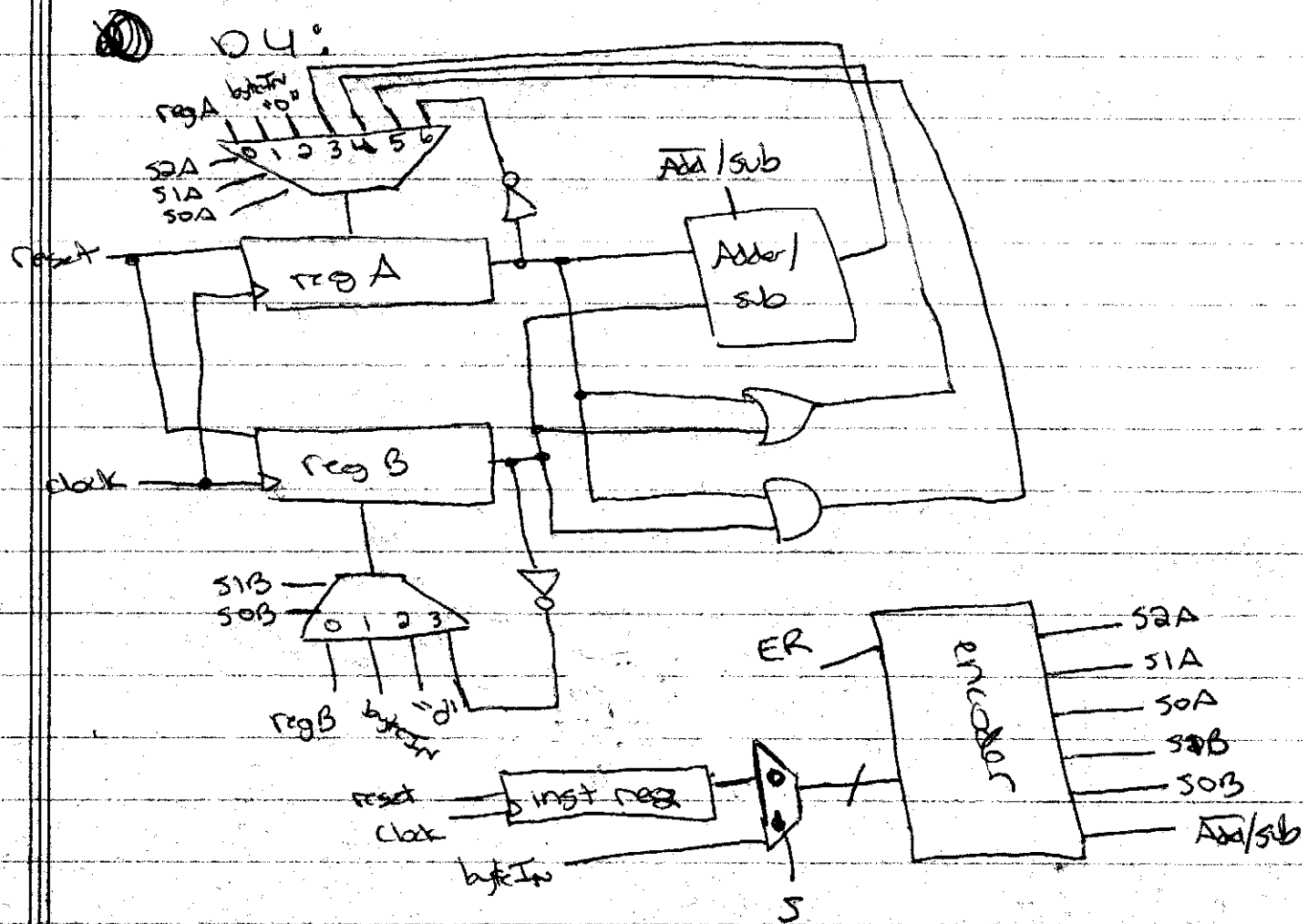
B) CU: determines whether or not the next byte is a piece of data or another instruction & controls the inst encoder
→ simplifies controller

DU: encodes the instruction to set the appropriate control bits, sends correct data to register, adds data, ANDs data, OR's data, INV data, & clears reg.

c) CU:


reset
reset
FF or FE
FF & FE
reset
load inst
ER, X
other inst
ER, S
FF & FE
FF or FE
FF or FE
reset
load data
ER, S
FF & FE

d) DU:


regA byteIn "0"
0 1 2 3 4 5 6
S2A
S1A
S0A
Add/sub
reset
regA
Adder/sub
regB
clock
S1B
S0B
0 1 2 3
regB byteIn "1"
ER
encoder
S2A
S1A
S0A
S0B
S0B
Add/sub
reset
inst reg
Clock
byteIn
S

Functionality of the encoder

functionality	ER	inst	S2A	S1A	S0A	S1B	S0B	Add/Sub
reset / hold A hold B	1	X	0	0	0	0	0	0
load A hold B	0	FF	0	0	1	0	0	X
hold A load B	0	FE	0	0	0	0	1	X
clear A hold B	0	FD	0	1	0	0	0	X
hold A clear B	0	FC	0	0	0	1	0	X
adder hold B	0	FB	0	1	1	0	0	0
sub hold B	0	FA	0	1	1	0	0	1
AND hold B	0	F9	1	0	1	0	0	X
OR hold B	0	F8	1	0	0	0	0	X
inv A hold B	0	F7	1	1	0	0	0	X
hold A inv B	0	F6	0	0	0	1	1	X
	else		0	0	0	0	0	X

note: the inst reg is to hold the inst for a load A or load B

e.g. for state "load inst":

ER=1, encoder is reset which will set all outputs =0 resulting in Reg A & Reg B holding their values

~~for state "load data"~~

at the same time, FF is placed on the input to inst reg

clock → state moves to "load data"

FF is in inst reg

~~→ encoder sees FF & places~~

$S = 0 \Rightarrow$ encoder sees the inst reg data, not the parallel input data

$\Rightarrow$ enables are set to place data on the input of the A reg