CDA 4102 -03 Bellenot FALL 85

HW #	Problems	due
1	Ch1 15, 17, 19 Ch2 9, 10, 11 Ch3 14, 16, 17	4 sept
2	Ch2 13, 15, 17 Ch3 21, 23, 33 Ch4 5, 14, 16	9 Sept
3	Ch2 18, 21 Ch3 5, 22 Ch4 10, 11, 12, 18, 25	16 sept
4	Ch2 1~ Ch3 26 Ch4 21, 28 Ch5 1, 2, 3, 6, 10	23 sept
5	Ch5 5, 7, 8, 13, 14, 15, 17, 20	30 sept test 1
6	Ch5 15, 16 Ch6 11, 12, 13, 22 Ch7 1, 2, 9	14 Oct
7	Ch7 6, 7, 17, 21, 28, 29, 30, 31, 32	21 Oct
8	Ch7 16, 18, 20, 22, 23, 24, 33, 34, 38	28 Oct
9	Ch8 2, 5, 8, 10a, 21, 24, 25	4 Nov test 2
10	Ch9 8, 9 Ch10 6, 7 Ch11 2, 3, 4, 16, 19	18 ~~Nov~~
11	Ch11 5, 7, 9, 10, 12, 17, 20, 24, 41, 42, 43	25 Nov
12	Ch12 2, 5, 7, 8, 21, 22, 23, 25, 26	2 dec

Test dates

Final: 8pm Wed 11 dec.
Test1: in class Wed 9 Oct
Test2: in class Wed 13 Nov.

Bellenot's office hours MWF 1:25-2:15 218 LAE
usually around most afternoons

TP #	Subject	due
1	1 bit register	4 sept
2	3 bit register	11 sept
3	FIFO buffer 1	18 sept
4	FIFO buffer 2	25 sept
5	Delay circuits	2 Oct test1
6	Multication unit	16 Oct
7	Software Stack pointer	23 Oct
8	Micro Code [& 4 Pg of Notes]	30 Oct
9	Asyn Tx	6 Nov test2
10	Asyn Rc [& 4 Pg of circuits]	20 Nov
11	Rc Flags	27 ~~Nov~~
12	Joan's bike	4 dec

REG

TP1 4 SEPT 85
due ~~11 JAN 85~~

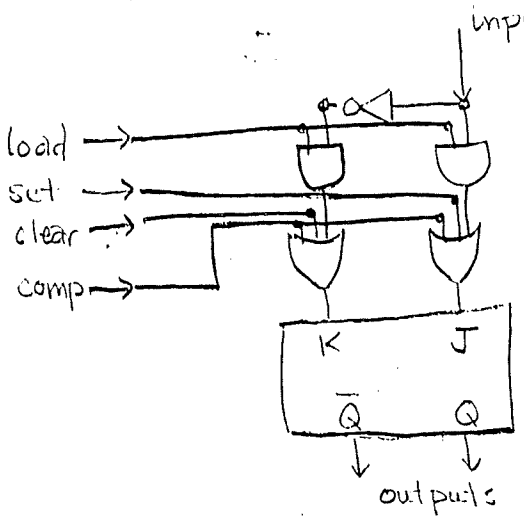
1-bit register R.

R has two outputs $(Q, \bar{Q})$, one input (for load) and four control lines (CLEAR, SET, LOAD, COMPLEMENT).

R satisfies the following table

input	control inputs: clear	set	load	compl.	current state	next state
y	0	0	0	0	Q	Q
y	1	0	0	0	Q	0
y	0	1	0	0	Q	1
y	0	0	1	0	Q	y
y	0	0	0	1	Q	$\bar{Q}$
y	← all others →				Q	? (doesn't matter)

For example R could be realized using a JK flip flop via the following circuit



R.

YOUR PROBLEM: WITHOUT puting any gates on the clock line NOR using preclear or preset, realize the 1-bit register R

A. with a D flip flop

B. with a T flip flop

TP 2 due ~~24 Jan~~ 11 Sept 1985 3-bit register R

R has 6 Outputs $R_2, R_1, R_0, \overline{R}_2, \overline{R}_1, \overline{R}_0$ and 3 inputs y_2, y_1 and y_0. There are also control lines for

Load : $R \leftarrow y_2 y_1 y_0$
clear : $R \leftarrow 0$
increment : $R \leftarrow R+1$
decrement : $R \leftarrow R-1$
two's complement : $R \leftarrow$ the 2's complement of what is in R

Draw a circuit that realizes the register R subject to the following conditions

1. Use T flip-flops (3)
2. Do not put any gates on the clock line (which can be implicit)
3. Do not use preclear or pre set
4. Do not use any adders (full or half)
5. When all control lines are at 0, the contents of R doesn't change
6. If two or more of the control lines are at 1, we are in a "don't care" mode

Hint: The following algorithm yields the 2's complement of the binary number $b_{n-1} b_{n-2} b_{n-3} \cdots b_3 b_2 b_1 b_0$ in an n-bit register.

```
i ← 0
while b_i = 0 do i ← i+1
i ← i+1
while i ≤ n do { b_i ← b̄_i
                 i ← i+1
```

This algorithm can be done in one clock pulse.

TP 3 & 4 A sort of FIFO buffer designed by two people mad at each other or by a schizoid. X, A, B, C, D, E are registers (n-bits wide, where n is large enuff to avoid overflow in this problem). And a, b, c, d, e are 1 bit flags. The micro-operations controlling this buffer are (note: no timing)

$1: X \leftarrow X+1$

$a'+b'+c': A \leftarrow X, a \leftarrow 1$

$b'+c': B \leftarrow A, b \leftarrow 1$

$bc': C \leftarrow B, c \leftarrow 1$

$cd': D \leftarrow C, d \leftarrow 1, c \leftarrow 0$

$de': E \leftarrow D, e \leftarrow 1, d \leftarrow 0$

TP 3 due ~~31 JAN~~ 18 sept 85

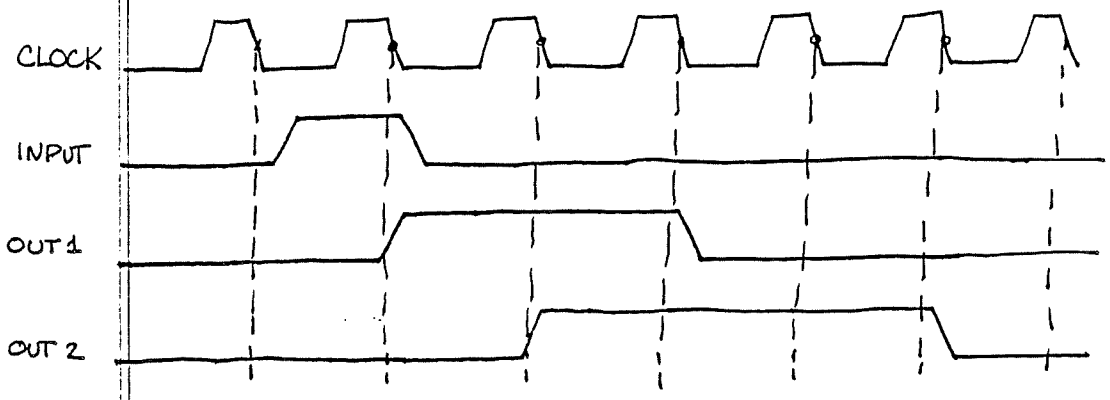
given the μ-operations above and the starting contents below continue filling in the table until all registers but X have stopped changing

	X	A	B	C	D	E	a	b	c	d	e
starting values	20	10	8	6	4	2	0	0	0	0	0
clock pulse per line											

TP 4 due ~~12 FEB~~ 25 sept 85

using RS-flip-flops for the flags a, b, c, d, and e draw a block diagram for this circuit. (ie the control logic the executes these μ-op's)

TP#5 due ~~21 Feb~~ 20 Oct 85 delay circuits



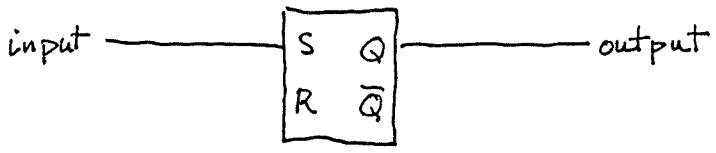
Design two circuits one which takes the given input and produces OUT1. The other takes the given input and produces OUT2.

There are, of course, many ways to do this and what is wanted is a simple solution. The simplest solution is not required. But the ~~more~~ farther your result is from the simplest ~~could~~, the farther your grade will be from prefect. Your circuit should not contain "hazards"!

partA. The simplest circuit which from input produces OUT1

partB The simplest circuit which from input produces OUT2

partC Again the circuit which from input produces OUT1 but this time it must include



6 16 Oct

TP ~~X~~ Due ~~18 Apr~~ 1985 ORG

An integer Multiplication Unit.

X & Y are 8-bit registers. T & PP are 16-bit registers

Z is a D flip flop. P is a 1-bit flag. (P=1 means X is a positive 8-bit number, P=0 means X is in 2-complement)

CC (cycle counter) is 3 RS flip-flops which when followed by a 3x8 decoder produces the controls

$C_0, C_1, C_2, \ldots$ [reg. numbered from right starting w/zero]

C_0: idle

C_1: not used until part C

C_2: (same clock pulse)
- PP ← 0,
- shr Y Z (i.e $Y_7 \leftarrow 0$, $Z \leftarrow Y_0$),
- if (P=1) then ($T(7-0) \leftarrow X$ and $T(15-8) \leftarrow 0$),
- if (P=0) then ($T(7-0) \leftarrow \overline{X}$ and $T_{15} \leftarrow \overline{X}_7$ and $T_{14} \leftarrow \overline{X}_7$ and $T_{13} \leftarrow \overline{X}_7$ and ... $T_8 \leftarrow \overline{X}_7$),
- CC ← 3

C_3: (same cp)
- if (Z=1) then PP ← PP + T
- shr Y Z ($Y_7 \leftarrow 0$, $Z \leftarrow Y_0$)
- shl T ($T_0 \leftarrow 0$)
- if Y=0 then CC ← 4

C_4: nothing

A. Suppose X contains -12 (2-comp), Y contains 5 and CC=2 show the contents of each register and each flag in binary after each clock pulse until they stop changing, NOTE P=0

B. Draw a block diagram show all control lines.

C. This will not work if Y is a negative number in 2's complement but if Y<0 we can multiply both X & Y by -1. Write down this in Reg Trans lang using C_1.

23 Oct

TP 7 Software Stack Pointer due ~~28 Feb 85~~

In Ch7 we have seen several uses for a register used as a "stack pointer" to a stack in memory. However the Ch 5 computer does not have such a register. Your job is to add a stackpointer via software (as in Ch6).

Some storage is reserved to help you

SP:	(the stack pointer)
OLD_AC:	(temp storage for AC)
SUB_ADD:	(temp storage for subroutine address in call)
TEMP:	(additional storage)

Write the routines (SP grows to high memory; AC protected)

A. Pop AC ← TOS
B. Push TOS ← AC
C. Call TOS ← PC, PC ← SUBADD {explain
D. Return PC ← TOS
E. 0-address ADDITION PUSH (POP + POP) (sortof)

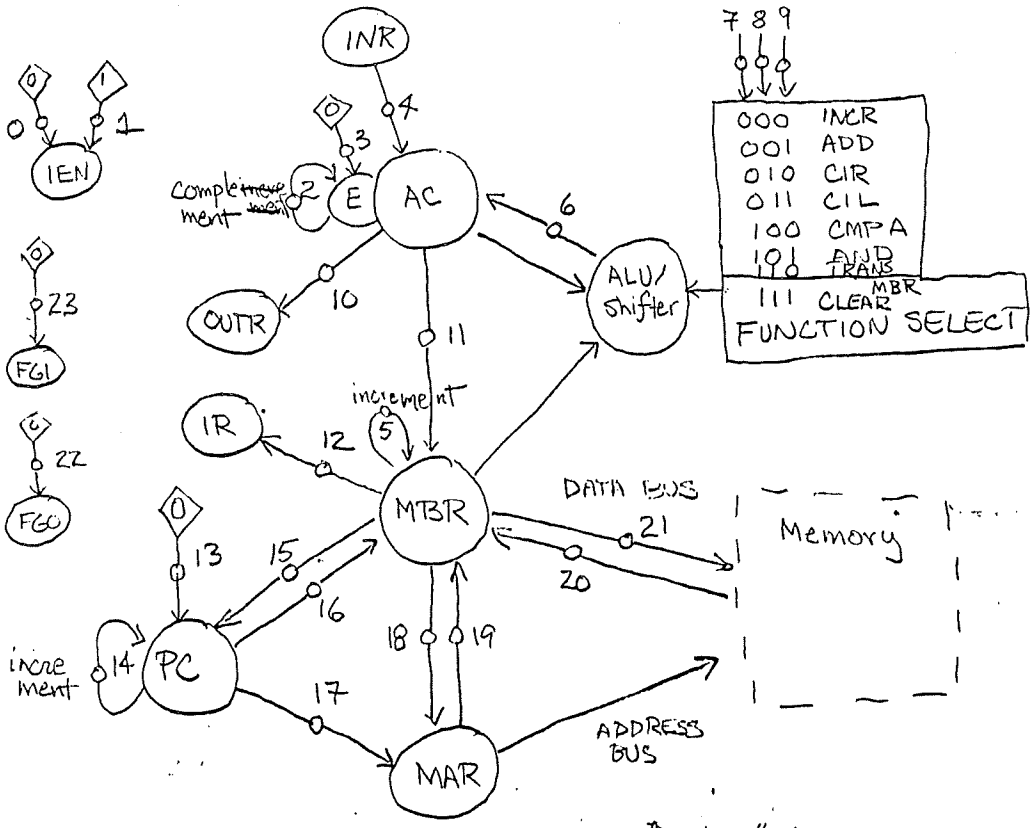
using the Ch.6. notation.

DATA FLOW / MICRO CODE
(or Hardcore Software)

ORG

Our object is to design a computer that will generate the control for the Chapt. 5 computer.

1. An Alternate view of the Ch. 5. Computer



We need a total of 24 "control valves" (which are numbered 0 to 23). Once again we can turn this into a control word which we will write in Hex. Examples

02**0**000 just turns on valve 17 i.e. MAR ← PC
104000 valves 20 & 14 : MBR ← M, PC ← PC+1
001000 valve 12 IR ← MBR(OP&I)

We have just done a fetch.

To make valve 17 work like this is easy. It needs only to select line 17 over line 18 at the input of MAR and put a "1" on the LOAD control for the MAR.

The width of each line differs: Line 10 is 8 bits wide, Line 11 is 16 bits, Line 6 is 16 bits when "7" = 1 and 17 bits when "7" = 0.

The diamonds contain values: line 0 when selected does IEN ← 0, while line 1 does IEN ← 1.

Not all control words are valid. At most one of "0" or "1" can be selected. Similarly for 16, 19, 20 & 11

2. A very simple computer A CONTROLER

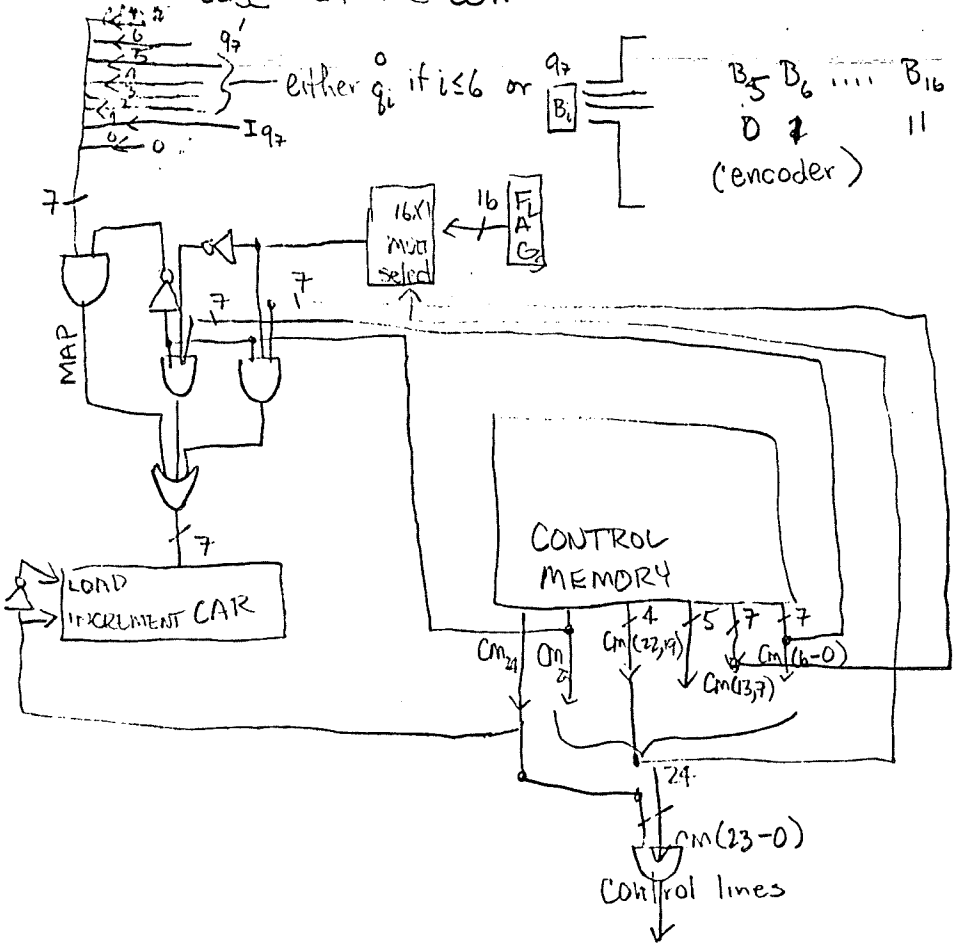
Our Control Computer has a 7-bit CAR (Control Memory Address register) hooked to a very fast ROM of 2^7 - 25 bit words. It has 1 Output ~~Many~~ namely the 24 control lines. It has 2 inputs the FLAGS which give the status of the chapter 5 computer and MAP which translates OPCODES of the chapter 5 computer into addresses for the CAR.

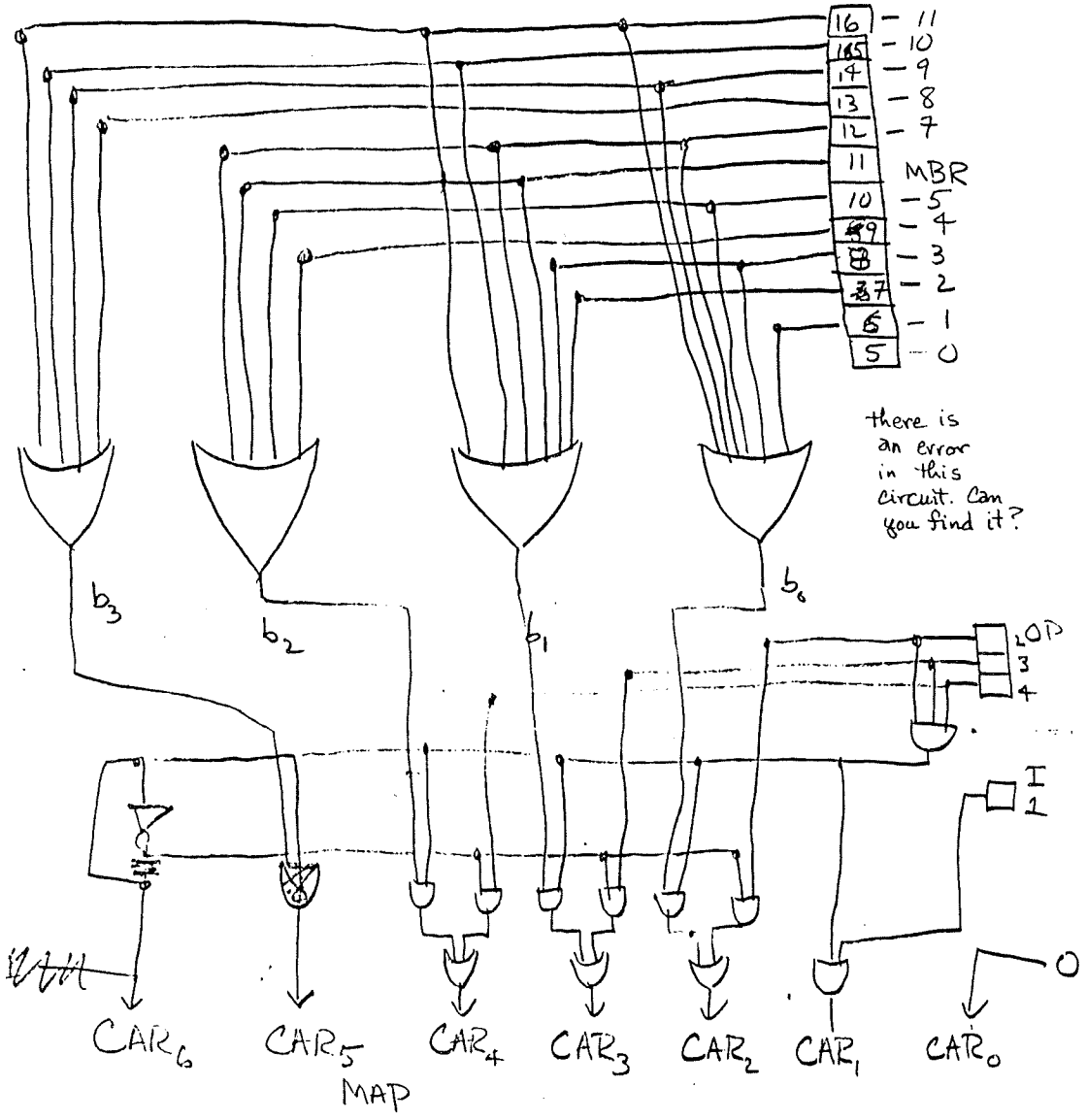
Basically it has two instructions; OUT with $CM_{24} = 1$ (CM = Control Memory [CAR]) which just outputs CM(23,0) for control, and a conditional jump $CM_{24} = 0$. This Jump takes ~~two~~ forms:

$CM_{23} = 1$ Field Cm(22-19) chooses a flag if it is 1
CAR ← CM(13-7) otherwise CAR ← CM(6-0)

$CM_{23} = 0$ CAR ← MAP

in either case all the control lines are zero.





FLAG	
0	MBR≠0 (set if MBR isn't zero)
1	FG1
2	FG0
3	AC(1)
4	AC=0
5	E
6	Interrupt
7	I (indirect bit) and not q_7

MICRO CODE

	00	01	10	11
00000	AND: CW(18)	SKIP: CW(14)	CLA: CW(6,7,8,9)	SNA: CJMP3 SKIP,CHEC
00001	CW(20)	CHECK: CJMP(6) RUP, FETCH	UJMP CHECK	NU
00010	CW(6,7,9)	FETCH: CW(17)	INP: CW(4,23)	NU
00011	UJMP CHECK	CW(20,14)	UJMP CHECK	NU
00100	ADD: CW(18)	CW(12)	CLE: CW(3)	SZA: CJMP4 SKIP CHEC
00101	CW(20)	CJMP(7) IND, EX	UJMP CHECK	NU
00110	CW(6,9)	EX: JMP MAP	OUT: CW(10,22)	NU
00111	UJMP CHECK	IND: CW(18)	UJMP CHECK	NU
01000	LDA: CW(18,6,7,8,9)	CW(20)	CMA: CW(6,7)	SZE: CJMP5 CHECK SKI
01001	CW(20)	JMP MAP	UJMP CHECK	NU
01010	CW(6,9)	RUP: CW(16,13)	SKI: CJMP(7) SKIP, CHECK	NU
01011	UJMP CHECK	CW(17,14)	NU	NU
01100	STA: CW(18)	CW(21,0)	CME: CW(2)	HLT: Blow Fuse
01101	CW(11)	UJMP FETCH	UJMP CHECK	(what should
01110	CW(21)		SKO: CJMP(2) SKIP, CHECK	happen here.
01111	UJMP CHECK		NU	
10000	BUN: CM(15)		CIR: CW(6,8)	
10001	UJMP CHECK		UJMP CHECK	
10010	NU		ION: CW(0)	
10011	NU		UJMP CHECK	
10100	BSA: CW(18,15,16)		CIL: CW(6,8,9)	
10101	CW(21)		UJMP CHECK	
10110	CW(14)		IOF: CW(1)	
10111	UJMP CHECK		UJMP CHECK	
11000	ISZ: CW(18)		INC: CW(6)	
11001	CW(20)		UJMP CHECK	
11010	CW(5)		NU	
11011	CW(21)		NU	
11100	CJUMP(0) CHECK, SKIP		SPA: CJMP3 CHECK, SKP	
11101			NU	
11110			NU	
11111			NU	

30 Oct

TP 8 due ~~7 Mar 85~~ μ-Code ORG

1 A. Change the micro code on Pg 4 so that STA has op code 9_1 and ADD has op code 9_3

B. Add the instruction NOP (no operation) with op code $I9_7B_{11}$ by changing the contents of one address in control memory (which address?)

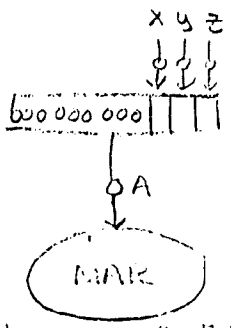
2. It is decided to replace ISZ with DSZ (decrement and skip if zero). Someone suggested changing value "5" from increment to decrement and not touching the micro-code.

A. Will this in fact change ISZ to DSZ?

B. Will this change have any unwanted side effects in terms of the micro control? why or why not?

3. By just changing the content of location 0101001 in control memory and with addition of FLAG 8 = MBR_1 (left most bit) change the computer to allow "infinite" indirection. i.e. if the left most bit is 1 then its address part is a pointer and not the effective address. Thus we stay in the indirect cycle until the left most bit is 0.

4. Add the figure to right to figure on page 1. If A=1, then MAR ← $(xyz)_2$ i.e. if x=1, y=0, z=1 and A=1 then MAR ← 5. Also add the values A, x, y & z to our control word. Using the micro-code format of page 1 (i.e. CW(11,21))



A. Write a triple of control words which does "M[3] ← M[6]"

B. Write a triple of control words which does "BUN 7" (no indirection)

[you may assume the triple is followed by UJMP Check.]

5. Still using the setting of problem 4 write a sequence of six control words which does "M[0] ← M[7] + M[4]" (do NOT protect the value in AC).

TP9 due ~~28 March~~ 6 Nov 85 Asyn Transmitter

TX is a 10 bit register with TX_9 connected to a tristate buffer gate which is turned on and off by the RS flip-flop LE (line enabled). The output of the tristate gate is hooked to the LINE (also used in TP's 10 & 11).

TB is 8 bits wide, Count is 4 bits wide and TBE is a flag (RS flip-flop). We have the following μ-ops

$$LE'TBE': \begin{cases} TX_9 \leftarrow 0,\ TX(8-1) \leftarrow TB,\ TX_0 \leftarrow 1 \\ LE \leftarrow 1,\ TBE \leftarrow 1,\ COUNT \leftarrow 16_{10} \end{cases}$$

LE : if count = 0 then (Shl TX , $TX_0 \leftarrow 0$)

LE : if count ≠ 0 and TX(8-0) = 0) then LE ← 0

LE+TBE : count ← count − 1.

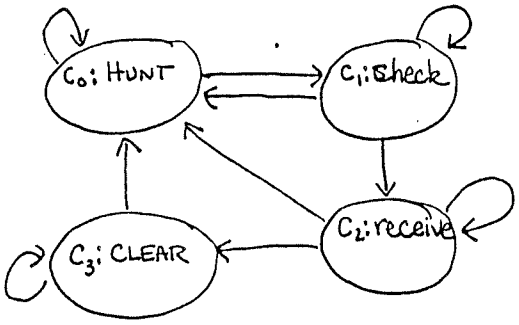
Draw a block diagram ~~for~~ circuit that realizes this Transmitter

Show how to modify this circuit so that TX_8 is loaded with an even parity bit instead of TB_7 when LE'TBE'

(TBE = transmit buffer empty)
(TX = transmitter)

TP 10 & 11 Asyn Receiver.

The following 4 pages are the same. They show the circuit of an Asyn Rc. Input (Serial) comes from 'LINE'. FGI & INR are as in the Ch.5 computer. There are 4 states



TP 10 due ~~4 Apr~~ 20 Nov

A. Using the circuit give the μ-ops for each of the states.

B. In words describe what each state is "doing" and what causes the state transitions.

Hint: "Color" the active lines a page per state.

TP 11 due ~~4 Apr~~ 27 Nov 1985 Receiver Flags

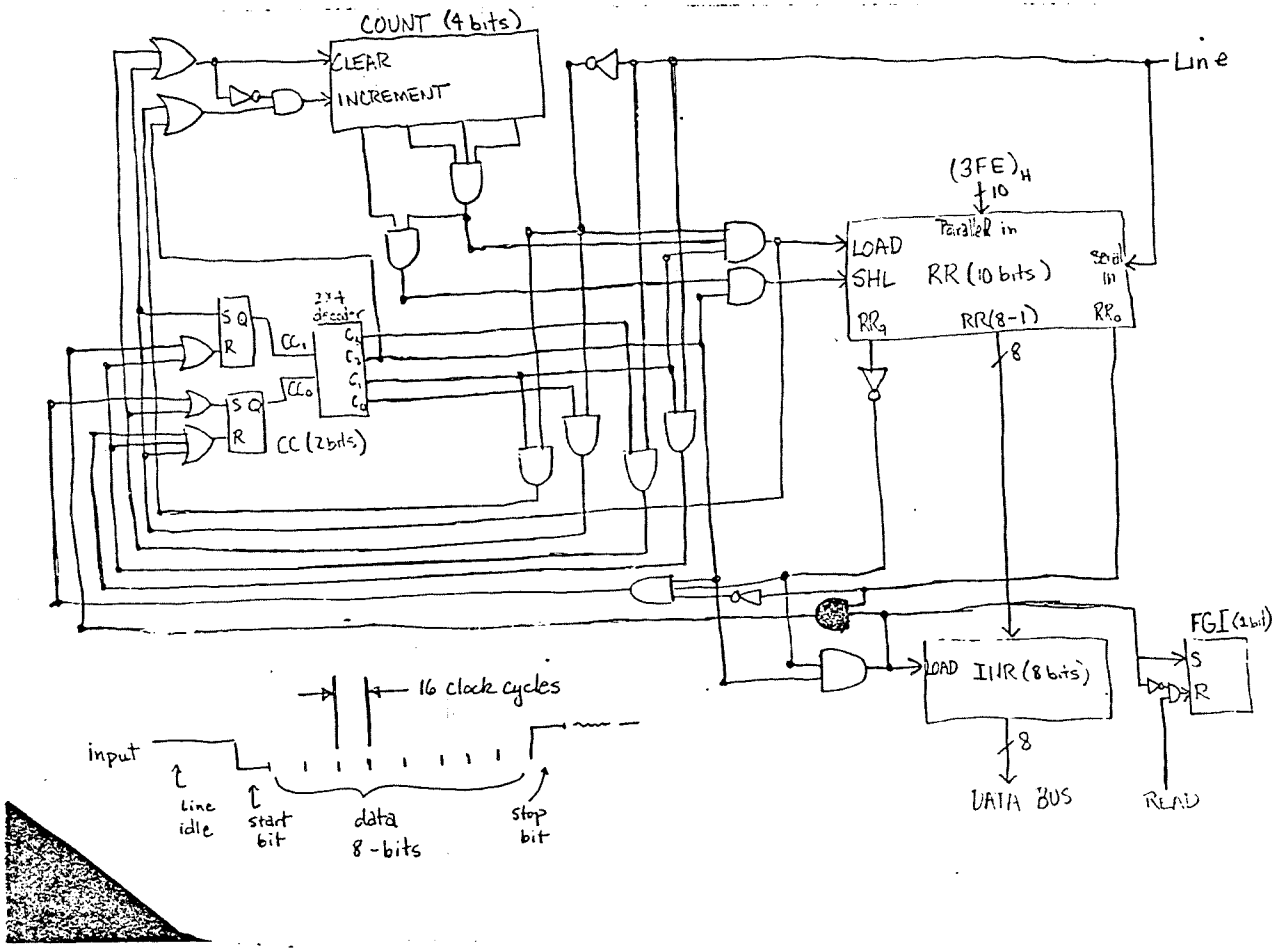
We want to add some error flags to our receiver. ER, E/$\bar{O}$ (even/odd), P/$\overline{NP}$ (parity/no parity) are control flags. FE (framing error), PE (parity error) and VE (~~overerror~~ overflow error) are status flags. The following μ-op's are added. (ER = error reset)

$C_2 RR_q' RR_0'$: FE ← 1
$C_2 RR_q' RR_0$ ER: FE ← 0
$C_2 RR_q'$ P: if parity doesn't match then PE ← 1
$C_2 RR_q'$ ER: if parity match then PE ← 0
$C_2 RR_q'$ FGI: VE ← 1
$C_2 RR_q'$ FGI'ER: VE ← 0

(match means same parity as E/$\bar{O}$)

Use RS flip-flops for FE, PE, VE and draw a circuit which does these μ-ops. You have only one 'xor' gate (with as many inputs as you need) and no 'xnor' gates.

CORRECTED 10 Apr 1985



15&16 Designing a digital bicycle speedometer: The idea is to count revolutions of the front wheel (in rev/sec) and translate this into speed (in mph). The equation $r = 13s$ is helpful, where r = number of revolutions of front wheel per second and s = speed in miles per hour.

A. Assume the register AR contains the number of revolutions in the last second. If speeds of at least 100 mph need to be accurately read, how wide must AR be? _____

BC. Assume the register C contains the total number of revolutions since the speedometer was last turned on. If distances of at least 1000 miles need to be accurately reported, how wide must C be? _____

D. To get the correct value in AR from the value in C we add registers OC (old C) and ROC (Real Old C) the same width as AR. Assume the control line p is "1" for exactly 1 clock cycle per second, then the micro-op p': ROC ← OC, OC ← C (the correct number of low order bits) is ~~done~~ added to the system. Explain how the value for AR can be obtained.

EFG Draw a block diagram of the system so far (when do you load AR?)

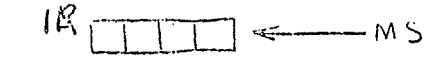
HI A "look up" table is used to translate the value in AR to the correct output codes to display the current speed. The output display has 3 digits (like 37.2) and each digit is displayed by a 7-segment display unit (decimal point is always on) What size (ie $2^n \times m$) ROM is needed? _____

JK. We are almost done with output, but the display units use too much of the battery an so we need a control line q which is "1" for exactly 1 clock cycle somewhere between 50 and 100 times a second. The clock rate is 1 Mega Hz and we add another register D with μ-op 1: D ← D+1 and let q be the boolean value of (D=0). Find the width of D _____ (values somehow will only be displayed when D=1, saving lots of current (i.e. battery juice))

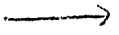
LMiN Finally we start on input. A tiny magnet is put on one spoke of the front wheel and a magnetic sensor is placed on the front fork. Spinning the front wheel, we find that the sensor "feels" the magnetic field for 1/32 of every rotation of the wheel. The clock rate is 1 MHz, how many clock pulses does the sensor "feel" the magnetic field each revolution if the bike is going 100 mph? _____ if the bike is going 1 mph? _____

OP If the sensor sends "logic 1" to the speedometer when it "feels" the magnetic field and "logic 0" otherwise, This input is serially loaded into the register IR which is four bits wide. label the sensor output line MS (magnetic sensor) write the μ-op which does this and include "control" part in the μ-op.

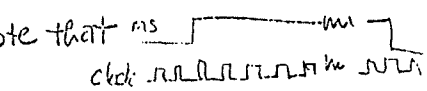
QRSTUVWXY:



Draw a circuit which translates the "leading edge" of MS into a 1 clock pulse long output (i.e. so that out: C ← C+1 keeps the correct count of the number of revolutions)



Note that



once Q=1 & IR=1111 out is "1" for one clock cycle then Out is zero until the next leading

Za: There are 2 states (Q=0) and (Q=1) describe what each is looking for

(Q=0):

(Q=1):

b Now we are ready to initialize the speedometer, which of our registers need to be initialized (AR, C, OC, ROC, D, IR) if we don't care if it gives incorrect speeds or distances for 30 secs or so but it needs to be correct after 1 min of operation

cd How would you initialize the value of C?