Edward Prokopik
CSC 137
HW 2

5-13

SUB:

D2T4: DR <- M[AR]

D2T5: DR <- AC, AC <- DR

D2T6: AC <- AC'

D2T7: AC <- AC + 1

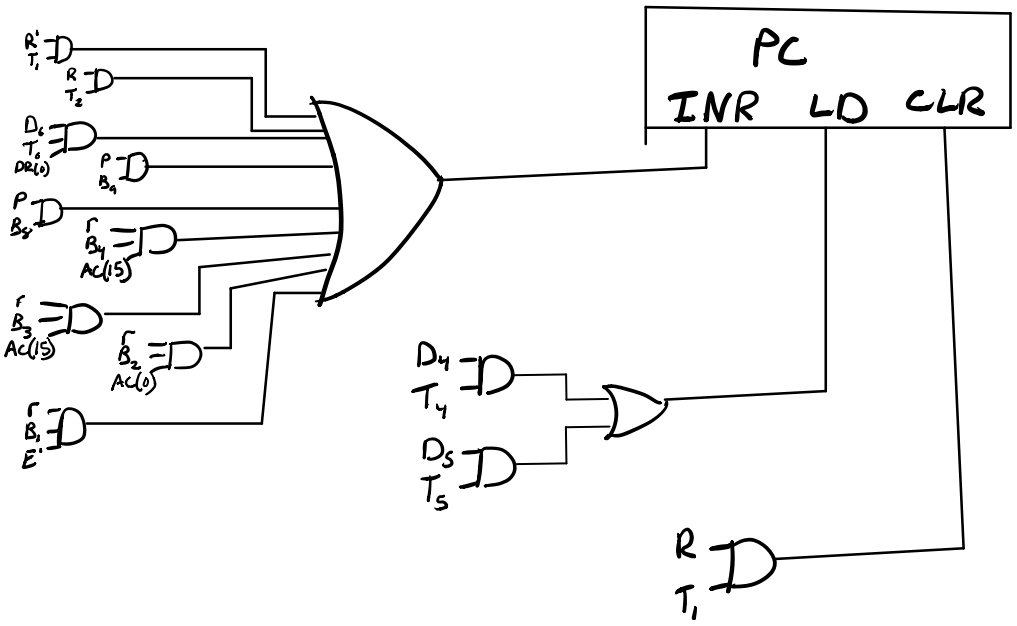
D2T8: AC <- AC + DR, SC <- 0

XCH:

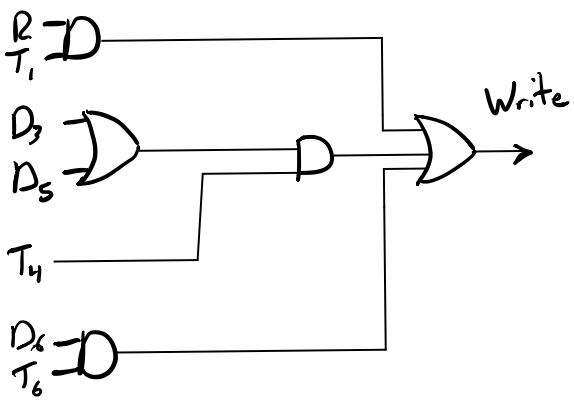
D3T4: DR <- M[AR]

D3T4: M[AR] <- AC, AC <- DR, SC <- 0

5-21



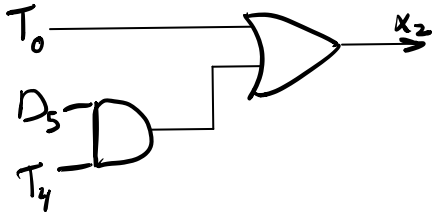
5-22



5-24

Register: PC

X2 = R'T0 + RT0 + D5T4 R' + R = 1
X2 = T0 + D5T4



1) Fetch and Decode

Fetch:
R'T0: LD(AR) X2
R'T1: INR(PC), LD(IR), READ, X7
Decode:
R'T2: LD(AR), LD(I), X5
Indirect:
D'7IT: LD(AR) READ, X7

INR(PC)	R'T1
LD(AR)	R'T0 + R'T2 + D'7IT3
LD(1)	R'T2
LD(IR)	R'51
READ	R'T1 + D'7IT3
X2	R'T0
X5	R'T2
X7	R'T1 + D'7IT3

2) Memory Reference Instructions

AND:
D0T4: LD(DR), READ, X7
D0T5: CLR(SC), AND, LD(AC)

ADD:
D1T4: LD(DR), READ, X7
D1T5: CLR(SC), ADD, LD(AC), LD(E)

LDA:
D2T4: LD(DR), READ, X7
D2T5: CLR(SC), X3, LD(AC)

STA:
D3T4: WRITE, X2, INR(AR)

BUN:
D4T4: CLR(SC), LD(PC), X1

BSA:
D5T4: INR(AR), WRITE, X2
D5T5: CLR(SC), LD(PC), X1

ISZ:
D6T4: LD(DR), READ, X7
D6T5: INR(DR)
D6T6: CLR(SC), if DR=0 then INR(PC), WRITE, X3

Alphabetically:

ADD:	D1T5
AND:	D0T5
CLR(SC):	(D0 + D1 + D2 + D5)T5 + (D3 + D4)T4 + (D6)T6
INR(AR):	D5T4
INR(DR):	D6T5
INR(PC):	D6T6 and DR = 0
LD(AC):	(D0+D1+D2) T5
LD(DR):	(D0 + D1 + D2 + D6) T4
LD(E):	D1T5
LD(PC):	D4T4 + D5T5
READ:	(D0 + D1 + D2 + D6) T4
WRITE:	(D3 + D5) T4 + D6T6
X1:	D4T4 + D5T5
X2:	D5T4
X3:	D2T5 + D6T6
X4:	D3T4
X7:	(D0 + D1 + D2 + D6) T4

3) Interrupt Handling

Interrupt handling is when a specific condition causes the computer to stop what it was doing and take care of an operation that comes in and return to where the PC was at .

It is useful for making efficient use of the computer. Instead of checking for a flag to be set whenever there is input or output that needs to be transferred, the computer can do other operations instead of constantly checking. Especially since input will come in much more slowly than the computer is able to check for it.