Talal Jawaid

Professor Weide Chang

CSC 137

3/12/2019

**Written Homework #2**

5.13

**SUB:**

D2T4: DR <- M[AR]

D2T5: DR <- AC, AC <- DR

D2T6: AC <- complement(AC)

D2T7: DR <- AC + 1

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

**XCH:**

D3T4: DR <- M[AR]

D3T5: AC <- DR, M[AR] <- AC, SC <- 0

5.21

*Increment(PC)* = R’T1 + RT2 + D6T6(DR)’ + rB4(AC (15))’ + rB3(AC (15)) +   rB2(AC)’ + rB1(E) + pB9(FGI) + pB8(FGO)

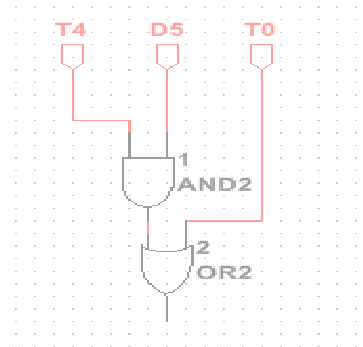
5.22

*Write* = RT1+ D3T4 + D5T4 + D6T6

5.24

X2 is PC on BUS

X2 = R’T0 + RT0 + D5T4 = (R’ + R)T0 +D5T4 = T0 + D5T4



**Three More Questions**

Towards the end of the chapter (p. 160-161), examples of deriving

control signals are given. Table 5-6 lists out the "actions" to

fetch/decode/execute each CPU instruction.

List out the control signals given when a CPU instruction is

being executed. For example, when each of the instructions X,

Y, and Z is being executed, the control unit gives a set of

signals at different time T:

X: at T4: A0 A1 C0

at T5, A1 B0

(function/inst X and at T4: signals A0 A1 C0 are given

function/inst X and at T5, signals A1 B0 are given)

Y: at T3, signals B1 C0

at T4, signals A1 B1 C1

Z: at T4: signals A0 C1

at T5, signals A1 B0 C0

Then, each signal is given when:

A0 = X T4 + Z T4 = (X+Z) T4

A1 = X T4 + X T5 + Y T4 + Z T5 = (X+Y) T4 + (X+Z) T5

B0 = X T5 + Z T5 = (X+Z) T5

B1 = Y T3 + Y T4 = Y (T3+T4)

C0 = X T4 + Y T3 + Z T5

C1 = Y T4 + Z T4 (Y+Z) T4

1. *For the Fetch, Decode, and Indirect steps (T0 through T3) list out signals like the above. First list out signals used in Fetch, Decode, then Indirect. Then for each signal, list its timing definition. Order the listing alphabetically. (There should be 8 different signals appeared 12 times.)*

*FETCH:*

R′T0: LD(AR), x2

R′T1: LD(IR), READ, INCR(PC), x7

*DECODE:*

R′T2: DECODE (IR(12 − 14)), LD(AR), x5

*INDIRECT:*

D7 ′IT3: LD(AR), x7, READ

*DECODE:* R′T2

INCR(PC): R′T1 + RT2 + R6T6 + rB4+ rB3 + rB2 + rB1 + pB9 + pB8

= T1 + T2 + R6T6 + r (B3 + B2 + B1 + B9 + B8)

LD(AR): R′T0 + R′T2 + D7 ′IT3 = R′(T0 + T2 ) + D7 ′IT3

LD(IR): R′T1

*READ:* R′T1 + D7 ′IT3

X 2: R′T0

X 5: R′T2

X 7: R′T1 + D7 ′IT3

2. *For signals used to execute 7 different memory-reference instructions, sort them out the same way as above. First by instruction name, then by signal name. Order the listing alphabetically.*

**ADD:**

D1 T4 : LD(DR), x7 , READ

D1 T5 : LD(AC), ADD, CLR(SC)

**AND:**

D0 T4 : LD(DR), x7 , READ

D0 T5 : LD(AC), AND, CLR(SC)

**BSA:**

D5 T4 : WRITE, x2 , INC(AR)

D5 T5 : LD(PC), x1 , CLR(SC)

**BUN:**

D4 T4 : LD(PC), x1 , CLR(SC)

**ISZ:**

D6 T4 : LD(DR), x7 , READ

D6 T5 : INC(DR)

D6 T6 : E, X3 , INC(PC), CLR(SC)

**LDA:**

D2 T4 : LD(DR), x7, READ

D2 T5 : LD(AC), x3 , CLR(SC)

**STA:**

D3T4 : WRITE, x4 , INC(AR), CLR(SC)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ADD = D1 T5

AND = D0T 5

CLR(SC) = (D0 + D1 + D2 + D5 )T5 + (D3 + D4 )T4 + D6 T6

LD(AC) = (D0 + D1 + D2 )T5

LD(DR) = (D0 + D1 + D2 + D6 )T4

LD(PC) = (T4 + T5 )D4

INC(AR) = D4 T4 + D5 T5

INC(DR) = D6 T5

INC(PC) = D6 T6

X 1 = D4 T4 + D5 T5

X 2 = D5T4

X 3 = D2 T5 + D6 T6

X 4 = D3 T4

X 7 = (D0 + D1 + D2 + D6 )T4

3. *What is interrupt-handling and why is it useful for operation of I/O devices?*

Answer:

Interrupt handling checks to see If the I/O instruction is needed. It is useful because if computers check OUTPR or INPR manually for input, it would be every millisecond, but the devices response time would be every second, which is a waste of time.