Quinn Roemer

Professor Chris Grove

CSC 137

24 March 2020

Homework #2

5-13:

EA is the effective address that resides in AR during time T_4. Subtraction must be done using the 2's complement of the subtrahend by complementing and incrementing AC. Give the sequence of register transfer statements needed to execute each of the listed instructions starting from timing T_4. The value in AC should not change unless the instruction specifies a change in its content. You can use TR to store the content of AC temporary or you can exchange DR and AC.

- SUB (AC <- AC – M[EA]):

  D_2T_4 : DR <- M[AR]

  D_2T_5 : TR <- AC, AC <- DR

  D_2T_6 : AC <- AC', DR <- TR

  D_2T_7 : AC <- AC + 1

  D_2T_8 : AC <- AC + DR, SC <- 0

- XCH (AC <- M[EA], M[EA] <- AC):

  D_3T_4 : DR <- M[AR]

  D_3T_5 : M[AR] <- AC, AC <- DR, SC <- 0

5.21:

Derive the control gates associated with the program counter in the basic computer.

- LD(PC) = $D_4T_4 + D_5T_5$
- INR(PC) = $R'T_1 + RT_2 + D_6T_6Dr' + rB_4(AC(15)') + rB_3(AC(15)) + rB_2AC' + rB_1E' + pB_9(FGI) + pB_8(FGO)$
- CLR(PC) = RT_1

5.22:

Derive the control gates for the write input of the memory of the basic computer.

- Memory Write = $RT_1 + D_3T_4 + D_5T_4 + D_6T_6$

5.24:

Derive the Boolean logic expression for X_2 (see Table 5-6). Show that X_2 can be generated with one AND gate and one OR gate.

- $X_2 = R'T_0 + RT_0 + D_5T_4$

Question 1:

For the Fetch, Decode, and Indirect steps (T_0 through T_3) list out signals like shown. First list out signals used in Fetched, Decode, then Indirect. Then for each signal, list its timing definition. Order the listing alphabetically.

Part A:

- Fetch:

  $R'T_0$: Ld(AR), X_2

  $R'T_1$: INR(PC), Ld(IR), Read, X_7

- Decode:

  $R'T_2$: Ld(AR), Ld(I), X_5

- Indirect:

  D'_7IT_3: Ld(AR), Read, X_7

Part B:

- INR(PC) = $R'T_1$
- Ld(IR) = $R'T_1$
- Ld(AR) = $R'(T_0 + T_2) + D'_7IT_3$
- Ld(I) = $R'T_2$
- Read = $R'T_1 + D'_7IT_3$
- $X_2 = R'T_0$
- $X_5 = R'T_2$
- $X_7 = R'T_1 + D'_7IT_3$

Question 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.

Part A:

- AND:

  D_0T_4: Ld(DR), Read, X_7

  D_0T_5: Ld(AC), Clr(SC)

- ADD:

  D_1T_4: Ld(DR), Read, X_7

  D_1T_5: Ld(AC), Ld(E), Clr(SC)

- LDA:

  D_2T_4: Ld(DR), Read, X_7
  D_2T_5: Ld(AC), Clr(SC)

- STA:

  D_3T_4: Write, Clr(SC), X_4

- BUN:

  D_4T_4: Ld(PC), Clr(SC), X_1

- BSA:

  D_5T_4: Write, INR(AR), X_2

  D_5T_5: Ld(PC), Clr(SC), X_1

- ISZ:

  D_6T_4: Ld(DR), Read, X_7

  D_6T_5: INR(DR)

  D_6T_6: Write, Dr'INR(PC), Clr(SC), X_3

Part B:

- $Clr(SC) = T_5(D_0 + D_1 + D_2 + D_5) + T_4(D_3 + D_4) + D_6T_6$
- $Dr'INR(PC) = D_6T_6$
- $INR(AR) = D_5T_4$
- $INR(DR) = D_6T_5$
- $Ld(AC) = T_5(D_0 + D_1 + D_2)$
- $Ld(DR) = T_4(D_0 + D_1 + D_2 + D_6)$
- $Ld(E) = D_1T_5$
- $Ld(PC) = D_4T_4 + D_5T_5$
- $Read = T_4(D_0 + D_1 + D_2 + D_6)$
- $Write = T_4(D_3 + D_5) + D_6T_6$
- $X_1 = D_4T_4 + D_5T_5$
- $X_2 = D_5T_4$
- $X_3 = D_6T_6$
- $X_4 = D_3T_4$
- $X_7 = T_4(D_0 + D_1 + D_2 + D_6)$

Question 3:

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

- Interrupt handling is essentially a hardware implementation of a branch and save instruction. It allows the basic computer to save an instruction and handle a special I/O instruction before returning to execute its saved instruction. This is useful because it handles a certain inefficiency with I/O devices. Instead of making the basic computer wait for an I/O device to finish transferring data, it instead executes instructions until the data is ready, where an interrupt is given, the data processed, and normal execution resumed.