EE312: Assignment V

Embedded Systems

October 22, 2020

Report the contents of the Flag Registers and Calculate T-States for the execution of each code.

1. Manually populate the address range X to $X+N-1$ and Y to $Y+N-1$ ($N = 2^m$). Here, $X+i$ and $Y+i$ ($i = 0, \ldots (N-1)$) respectively store the integer and fractional part of the $(i+1)^{th}$ number expressed in fixed point representation. Write a code for computing the average and median of $N = 8$ numbers. Use the subroutines *ADDFPNUM*, *DIVFPNUM* and *SORTASCFPNUM*.

2. Manually populate the address range P to $P+N-1$ and Q to $Q+N-1$ with respective integer and fractional parts of N fixed point numbers. Compute the first difference of this sequence and save the results in the address range X to $X+N-1$ and Y to $Y+N-1$ for their respective integer and fractional parts. The first difference is defined as $dx[n] = 0.5(x[n+1] - x[n-1])$. For a sequence of length N, $dx[0] = 0.5(x[1] - x[0])$ and $dx[N-1] = 0.5(x[N-1] - x[N-2])$. Use the subroutine *SUBFPNUM* and *DIVFPNUM*.

3. Manually populate the address range X to $X+9$ with two-digit BCD numbers. Also, set Y to $Y+4$ with values of q from $\{2, 3, 4, 5\}$. Convert the BCD numbers to hex values prior to their binary crossover. Generate the children solutions $c8h1[i]$ and $c8h2[i]$ by using the subroutine $[c8h1[i], c8h2[i]] = XOVER(p8h1@(X+i), p8h2@(X+9-i), q@(Y+i))$ ($i = 0, \ldots 4$). Convert the results of binary crossover to BCD format using the subroutine $[x2dBCD@A] = HEX2BCD(x8h@B)$. However, the solutions may be 2 or 3 digit and accordingly require two or

three nibbles to store them. Here, we go for a linked list for storing the resultant BCD numbers. Make sure that the last data block points to $0000H$, where we possibly do not want to visit. Use the subroutine $ADD\ DATABLOCK(xBCD@B, b, SL16, NL16)$ for creating the linked list with the results of crossover.

The list of necessary subroutines are as follows.

(a) $[x8h@B] = BCD2HEX(x2dBCD@A)$ – A two digit decimal number $x2dBCD$ is stored in BCD format at an address A. This subroutine converts $x2dBCD$ to its 8-bit hexadecimal equivalent number $x8h$. The result is stored at an address B.

(b) $[x2dBCD@A] = HEX2BCD(x8h@B)$ – A hexadecimal number $x8h$ is stored at an address A. This subroutine converts $x8h$ to its decimal equivalent $x2dBCD$ and stores the same in BCD format (output may require 16-bits). The result is stored at an address B (and $B + 1$, if necessary).

(c) $[c8h1@C, c8h2@D] = XOVER(p8h1@A, p8h2@B, q@E)$ – Subroutine for binary crossover of two 8-bit hexadecimal numbers $p8h1$ and $p8h2$ (both lesser than $(99)_{10}$) stored at addresses A and B respectively. The resultant numbers $c8h1$ and $c8h2$ are stored at addresses C and D respectively. The crossover point q is specified at an address E.

(d) $ADDDATABLOCK(xBCD@B, b, SL16, NL16)$ – Subroutine for adding a datablock to a linked list at the start address $SL16$. The linked list data block has four consecutive bytes DB_1,DB_2,NA_1,NA_2. A three-digit BCD number uses lower nibble of DB_1 and entire DB_2. Otherwise, a single or two digit BCD number uses only DB_2. The byte b indicates whether the BCD number is 8-bit ($b = 00H$) or 16-bit ($b = 01H$). Accordingly, $xBCD$ is collected from B (and may be also $B + 1$). The MSB of DB_1 is set to the value of b. The last two bytes NA_1, NA_2 are used to store the address $NL16$ of the next data block.

(e) $[cI8h@C_1, cF8h@C_2] = ADDFPNUM(aI8h@A_1, aF8h@A_2, bI8h@B_1, bF8h@B_2)$ – Subroutine for adding two fixed point numbers ($c = a + b$). Their integer ($aI8h$, $bI8h$) and fractional parts ($aF8h$, $bF8h$) are respectively stored at addresses (A_1, B_1) and (A_2, B_2). The integer ($cI8h$) and fractional ($cF8h$) of the result c are stored at C_1 and C_2 respectively.

(f) $[cI8h@C_1, cF8h@C_2] = SUBFPNUM(aI8h@A_1, aF8h@A_2, bI8h@B_1, bF8h@B_2)$ – Subroutine for subtracting two fixed point numbers $c = a - b$. Their integer ($aI8h$, $bI8h$) and fractional parts ($aF8h$, $bF8h$) are respectively stored at addresses (A_1, B_1) and (A_2, B_2). The integer ($cI8h$) and fractional ($cF8h$) of the result c are stored at C_1 and C_2 respectively.

(g) $[bI8h@B_1, bF8h@B_2] = DIVFPNUM(aI8h@A_1, aF8h@A_2, m@C)$ – Subroutine for Dividing a fixed point number a by 2^m (m stored at address C). The integer ($aI8h$) and fractional part ($aF8h$) are respectively stored at addresses A_1 and A_2. The integer ($bI8h$) and fractional ($bF8h$) of the result b are stored at B_1 and B_2 respectively.

(h) $SORTASCFPNUM(xI8START, xF8START, sxI8START, sxF8START, n8h@A)$ – Subroutine for sorting fixd point numbers in ascending order. The starting adrresses of the integer and fraction part of the input sequence are provided by $xI8START$ and $xF8START$ respectively. The sequence is available as byte array in consecutive memory locations. The sequence length $n8h$ is avilable at address A. The starting addresses of the integer and fraction part of the sorted sequence are $sxI8START$ and $sxF8START$ respectively.