17 December 2012 -- Computer Architectures self-test -- part 1/2

Name, Matricola ....

There are some algorithms to convert unsigned integer binary number into their decimal BCD representation. Besides these based on recurring divisions, there is one particularly interesting as it is easy to be migrated to hardware, i.e. the **Shift and Add-3 Algorithm**. Below, the algorithm for 8 bits binary to BCD conversion:

1. Shift the binary number left one bit.
2. If 8 shifts have taken place, the BCD number is in the Hundreds, Tens, and Units column.
3. If the binary value in any of the BCD columns is 5 or greater, add 3 to that value in that BCD column.
4. Goto 1.

Example 1: Convert hex "E" to BCD

Operation	Tens	Units	Binary
HEX			E
Start			1 1 1 0
Shift 1		1	1 1 0
Shift 2		1 1	1 0
Shift 3		1 1 1	0
Add 3		1 0 1 0	0
Shift 4	1	0 1 0 0	
BCD	1	4	

Example 2: Convert hex "FF" to BCD

Operation	Hundreds	Tens	Units	Binary	
HEX				F	F
Start				1 1 1 1	1 1 1 1
Shift 1			1	1 1 1 1	1 1 1
Shift 2			1 1	1 1 1 1	1 1
Shift 3			1 1 1	1 1 1 1	1
Add 3			1 0 1 0	1 1 1 1	1
Shift 4		1	0 1 0 1	1 1 1 1	
Add 3		1	1 0 0 0	1 1 1 1	
Shift 5		1 1	0 0 0 1	1 1 1	
Shift 6		1 1 0	0 0 1 1	1 1	
Add 3		1 0 0 1	0 0 1 1	1 1	
Shift 7	1	0 0 1 0	0 1 1 1	1	
Add 3	1	0 0 1 0	1 0 1 0	1	
Shift 8	1 0	0 1 0 1	0 1 0 1		
BCD	2	5	5		

It is required to write a 8086 assembly program to implement the binary to BCD conversion using the **Shift and Add-3 Algorithm**. In particular:

- Item 1: the input data is on 8 bits in binary and the output is on 3 digits BCD, i.e. 12 bits. The variables are:
  - INPUT_BIN   DB   ?
  - OUTPUT_BCD   DB   3 DUP (?)

  where each byte of the array OUTPUT_BCD (in little endian mode) stores one BCD digit, either in the lower or higher half of the byte.

- Item 2: the input data is on 8 bits in binary and the output is on 3 digits BCD, i.e. 12 bits. The variables are:
  - INPUT_BIN   DB   ?
  - OUTPUT_BCD   DB   2 DUP (?)

  where OUTPUT_BCD stores the lower two BCD digits, and the lower half of OUTPUT_BCD+1 stores the highest BCD digit

- Item 3: the input data is on 16 bits in binary and the output is on 5 digits BCD, i.e. 20 bits. The variables are:
  - INPUT_BIN   DW   ?
  - OUTPUT_BCD   DB   5 DUP (?)

  where each byte of the array OUTPUT_BCD (in little endian mode) stores one BCD digit, either in the lower or higher half of the byte.

17 December 2012 -- Computer Architectures self-test -- part 1/2

Name, Matricola ....

- Item 4: the input data is on 16 bits in binary and the output is on 5 digits BCD, i.e. 20 bits. The variables are:
  - o INPUT_BIN   DB   ?
  - o OUTPUT_BCD   DB 3 DUP (?)

  where each byte of OUTPUT_BCD and OUTPUT_BCD+1 stores two BCD digits, and the lower half of OUTPUT_BCD+2 stores the highest BCD digit

Points for each completed item (as uncompleted items could be not evaluated)

- Item 1. or Item 2. = up to 22 points;
- Item 1. + Item 2. = up to 25 points;
- Item 3. or Item 4. = up to 29 points
- Item 3. + Item 4. = up to 33 points;

(no other combination except these above will be considered)

It is not required to provide the optimal (shortest, most efficient, fastest, …) solution, but a working and clear solution. The input-output part is not necessary in the class developed solution, but its implementation is mandatory to be discussed at oral exam.

IMPORTANT NOTE:
None.

Please use carbon copy and retain one copy for home implementation and debug. Please provide your classroom submitted solution with several explanatory and significant comments. When coming to oral discussion, please mark on your "classroom" copy all modifications. Please also provide an error-free and running release of the solution, as well as with its printed list of instructions. Please consider that the above are necessary but not sufficient requirements to success the exam, since the final evaluation will be based on a number of parameters.

FAILURE TO ACCOMPLISH ALL PREVIOUS NECESSARY REQUIREMENTS WILL CAUSE NO-QUESTION-ASKED AND IMMEDIATE REJECTION.