# INEL 4206. EXAM. OCTOBER 24, 2012

**Problem 1**. Answer the following questions:

1.1. What is the difference between a memory mapped I/O system and an IO mapped I/O system? **ANSWER:**
*In the memory mapped I/O, directions for the ports and peripheral registers are part of the normal memory map, and no special instructions are needed for dealing with them. In the IO mapped case, they have different memory map and special instructions are used (IN, OUT) to access them*

1.2. What is the difference between the Von Neumann and the Harvard architectures?
**ANSWER:**
*In the Von Neumann architecture, program and data memories share the same address and data buses. In the Harvard architecture they have different buses*

1.3. What is the range of integers normally covered with 8 bits in unsigned binary convention and in two's complement convention?
**ANSWER:** *0 to* $2^8 - 1$*, that is, 0 to 255*

1.4. What does 10011001 represents in a) normal binary convention; b) two's complement convention; c) biased 64 convention; d) BCD convention?
**ANSWER:**
a) 128 + 16+8+1 = 153; b) -128 + 16 + 8 + 1 = -103; c) 153+64=217; d) 99.

1.5. How many different items are defined in the ASCII table, and why?
**ANSWER:** *128, because the codes use 7 bits and* $2^7$*=128*

1.6. What is an addressing mode?
**ANSWER:** *Is the way or syntaxis in which the operand indicates where to find the datum*

1.7. Define the immediate addressing mode.
**ANSWER:** *The datum is the operand itself*

1.8. The overflow flag in the SR indicates an overflow in an addition. What does that mean?
**ANSWER:** *That two operands of the same sign are added but a result with different sign is obtained*

1.9. Define the register addressing mode.
**ANSWER:** *The datum is the contents of the register*

1.10. If a continuous interval $[a, b]$ is encoded using $n$ digits. What is the resolution?
**ANSWER:** $\frac{b-a}{2^n}$

1.11. In a word with $n$ bits, $a_{n-1}a_{n-2}\ldots a_1a_0$, what is the series power that describes the decimal equivalent in two's complement convention?
**ANSWER:**

$$-2^{n-1}\,a_{n-1} + 2^{n-2}\,a_{n-2} + \ldots + 2a_1 + a_0$$

1.12. In the fetch-decode-execute cycle, at what moment is the PC already loaded with the address of the next instruction? **ANSWER:** *Right after the decode phase*

1.13. What are the components of the CPU?
**ANSWER:** *Hardware components: ALU, CU, Registers, BIL. Software components: addressing modes and instruction set.*

1.14. The data address 0304h is for datum 34AF2301h. List the physical addresses for each of the bytes in little endian convention.
**ANSWER:**

[0304h]=01h; [0305h]= 23h; [0306h]= AFh; [0307h]= 34h

1.15. Repeat previous question in big endian convention
**ANSWER:**

[0304h]=34h; [0305h]= AFh; [0306h]= 23h; [0307h]= 01h

1.16. Define the indirect register mode.
**ANSWER:** *@Rn: The datum is in memory, at address given by the register contents*

1.17. Define the sign extension operation.
**ANSWER:** *An n-bit signed number can be extended to m-bit number, with m>n, by filling the extra bits all equal to the sign bit*

1.18. If the SP register shows 0400h, and the operation of push and pop is done with byte size data, what will the contents of SP be after a pop operation?
**ANSWER:** SP = 0401h.

1.19. When comparing unsigned numbers by subtraction, A-B, which flag and with what value will tell us that A≥B?
**ANSWER:** *The carry flag. C=1 if A≥B.*

1.20. Describe the process involved in calling a subroutine and returning from it.
**ANSWER:**
*When the subroutine is called, the contents of the program counter PC is pushed onto the stack and then loaded with the address of the first instruction of the subroutine. When returning from the soubroutine, the PC is popped from the stack, retrieving the address of the instruction that follows the call.*

**Problem 2**. An engineer is designing an application that needs to measure the temperature from -145 degrees to +130 degrees. The engineer first needs to determine the minimum number of bits that are needed in order to convert the analog measurement into a digital quantity.

2.1. Assuming that only the integer part is needed, what is the minimum number of bits necessary?

**ANSWER**

*If only integers are taken, then 130 - (-145) = 275 items need to be identified. Since* $2^8 < 275 < 2^9$*, we need 9 bits*

**Another reasoning:** *Since the resolution is* $\Delta = 1$*, then n must satisfy [130 - (-145)]/*$2^n$ *= 1 or* $\log_2 275 = 8.106$ *Sinc n must be an integer, n = 9.*

2.2. An improvement is needed and now the design has to be able to measure not only the integer part but also the fractional part of the temperature. If an error of less than 0.01 is desired, then how many bits total would be needed?

**ANSWER**

*There are 130-(-145) = 275 integer subintervals. Each one will be divided in 100 subintervals, for a total of 27,500 subintervals that need to be identified.* $2^{14} = 16,384$ *and* $2^{15} = 32768$*; therefore, the engineer would need 15 bits*

**Another reasoning:** *Since the resolution is* $\Delta = 0.01$*, then n must satisfy [130 - (-145)]/*$2^n$ *= 0.01 or* $\log_2 27500 = 14.75$ *Sinc n must be an integer, n = 15.*

2.3. Someone came along and decided to partition the temperature range into equally spaced subintervals using 10 bits without considering parts 2.1 and 2.2 above. What is the resolution of this scheme? Which values are included in the $7^{th}$ subinterval and what is the binary representation for all values within this subinterval?

**ANSWER**

*The resolution is*

$$\Delta = \frac{130 - (-145)}{2^{10}} = 0.26856$$

.

*The lower bound of the subinterval 7 is*

$$-145 + 7 \times 0.26856 = -143.12$$

*Hence the subinterval is [-143.12, -143.12+0.26856)= [-143.12, -142.852]*

*All numbers in this subinterval are represented by 0000000111, or 007h*

**Problem 3**. Assume a 20-bit address bus and an 8-bit data bus, together with several 32KB memory modules, with each module being independently activated or deactivated. The modules can all share the least significant lines of the address bus to access internal registers, but only an activated module would be accessed by the data bus. The additional non shared address lines are used to activate the appropriate module.

3.1. What is the maximum number of modules that can be used?

**ANSWER:**
*Total Number of possible Addresses*= $2^{20}$=*1 M.*
*Number of addresses per module = 32K* = $2^{15}$
*Number of possible modules* = $2^{20}/2^{15} = 2^5$= *32 modules.*

3.2. If all the possible memory modules were used, what memory size would be available?

**ANSWER:**
*1M*

3.3. Assume that only four memory modules are used to build a memory segment starting at address 20000h. What is the effective size of address space being used and what is the last address in this space?

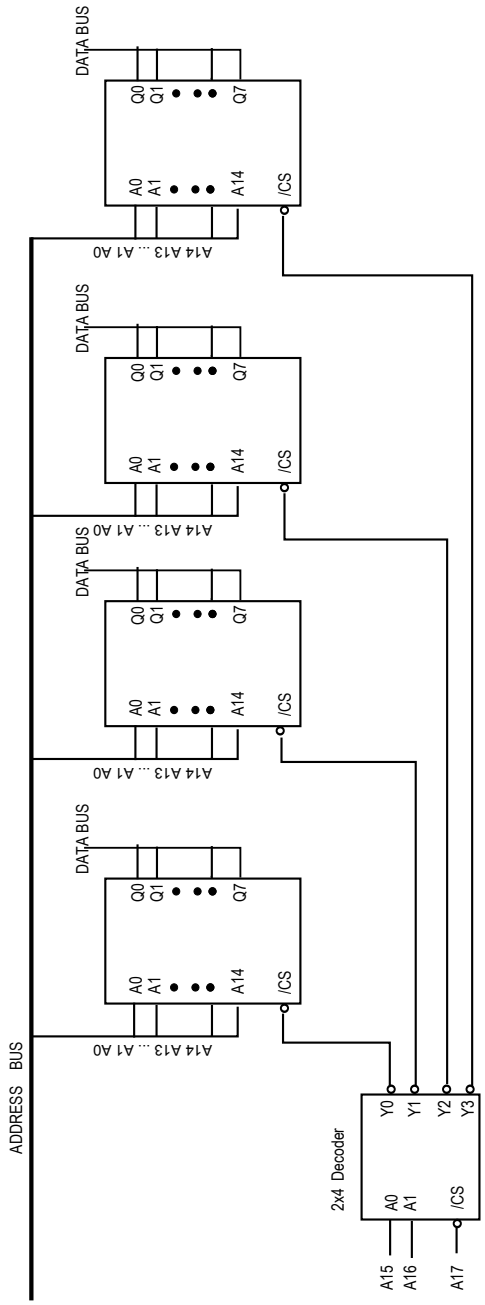
**ANSWER:**
*The efective size is* $4 \times 32KB$ = *128 KB.*
*128 K* = $2^{17}$*, using therefore 17 bits. The largest number with 17 bits is 1FFFFh. Hence the last address in this space is 20000h + 1FFFFh = 3FFFFh*

3.4. Since only four modules are being used, there are several ways in which the interface can be built. In the following page, design a logical scheme such that you do not need to use all the address lines.

3.5. For the interface designed in the previous item, is it true that from the CPU point of view each memory cell has a unique address? Justify your answer.[1ex]

**ANSWER:**
*NO, addresses are not unique. Since only 17 bits were necessary for address, leaving the three most significant bits for module activation. Since there are four modules, one* 2x4 *decoder is enough, requiring an extra bits for activation. Hence two bits are not, and correspond don't care values. Hence, from the CPU point of view, each location has four addresses. For example, address 20000h, 60000h, A0000h and E000h all point to the same location*

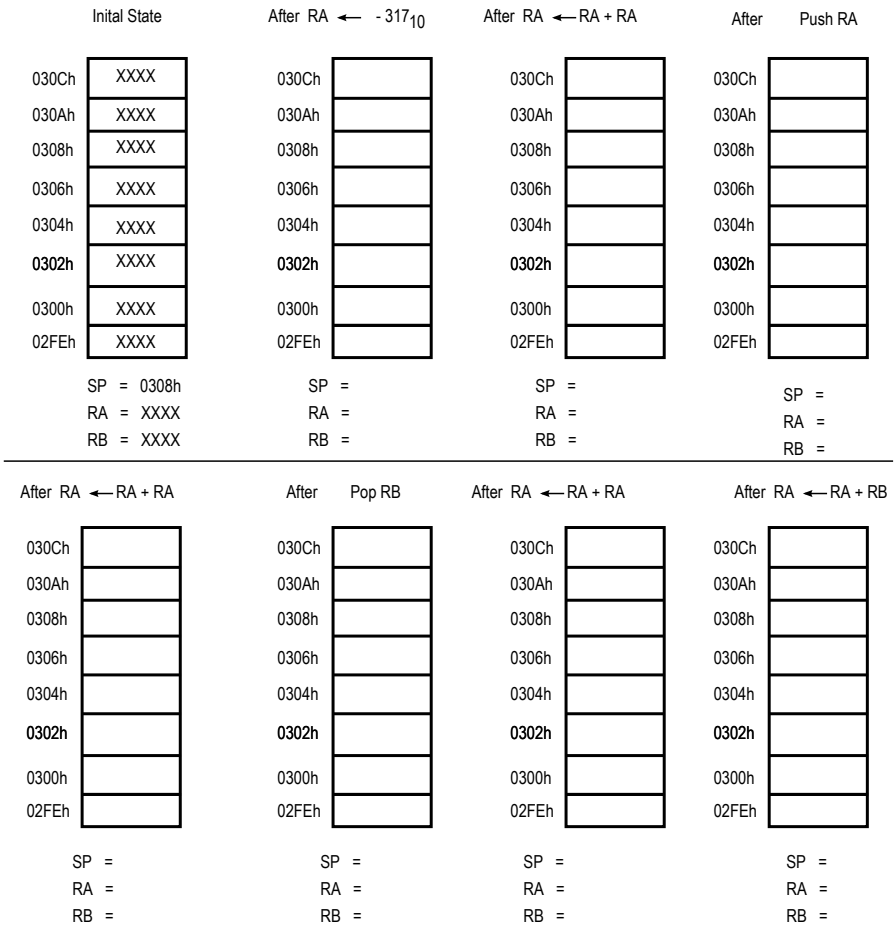
**ANSWER to item 3.4**:
*We need 15 bits for internal addresses in module, say A0 ... A14. Two bits to differentiate between modules, say A15 and A16. We can use a* 2x4 *decoder to activate the modules, requiring an extra bits for activation, say A17. Hence the two address bits A18 and A19 are not being used since we are looking for a minimum number of bits. A connection with these principles is shown below.*



**Problem 4**. Registers RA and RB are 16-bit wide. The Stack Pointer SP points at the last item that has been pushed and memory is denoted using data addresses for word size data. At certain moment SP = 0308h. The following sequence of instructions is given, where the number $-317$ is in decimal notation:

1: $RA \leftarrow -317$
2: $RA \leftarrow RA + RA$
3: Push RA
4: $RA \leftarrow RA + RA$
5: $RA \leftarrow RA + RA$
6: Pop RB
7: $RA \leftarrow RA + RB$

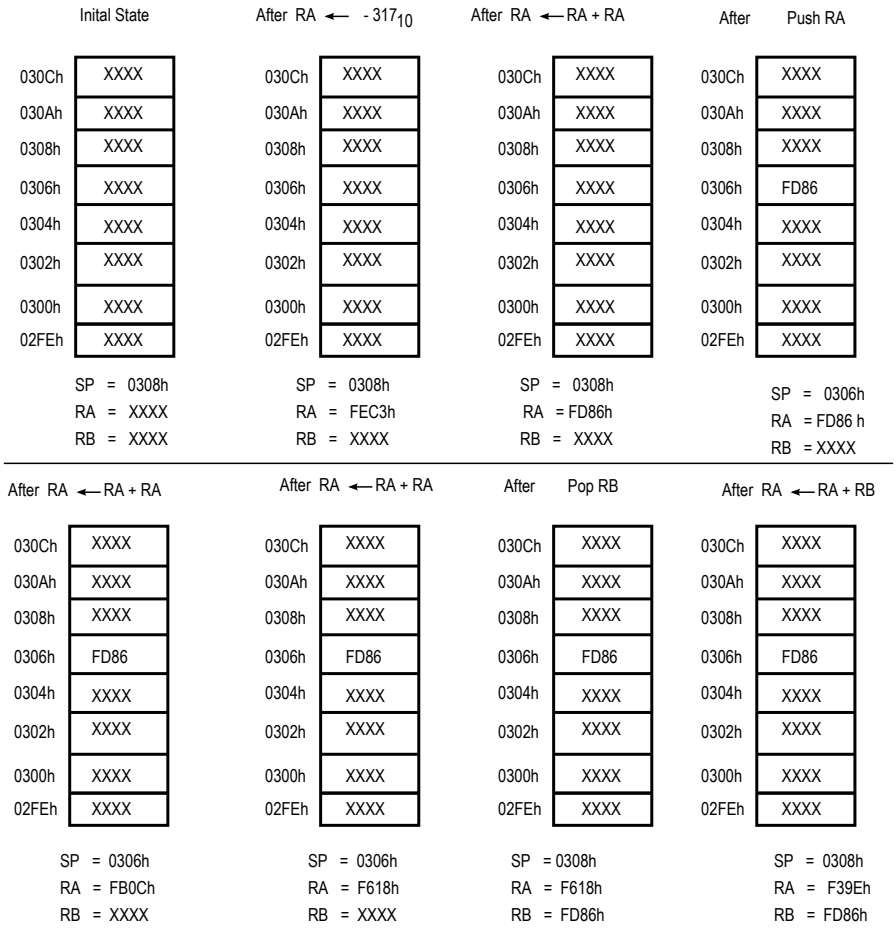
4.1. Fill the blanks for the contents of memory and the contents of registers in the figure below. Fill also those whose contents are unknown (you may use one X, for speed). All contents should be in **hex notation**.



**ANSWER:** *REMARK: By accident one of the steps was changed in order. The final result does not change, but the answer could be a little different depending on the order you considered* pop RB. *This answer will follow the figure. The difference is minimal.*

$-317 \rightarrow FEC3$ *the first step yielding* RA = FEC3.
FEC3h + FEC3h = 1FD86h, *and discarding carry we get* RA = FD86.
FD86 *is pushed to address* 0306h *and SP is decremented by 2:* SP= 0306h.
FD86h + FD86h = 1FB0Ch, *and discarding carry we get* RA = FB0C.
FB0Ch + FB0Ch = 1F618h, *and discarding carry we get* RA = F618.**
*The value pointed at by* SP=0306h *in stack is copied onto* RB, *and* SP *is incremented by 2, yielding* RB = FD86h, SP = 0306h.**
F618h + FD86h = 1F39Eh, *and discarding carry we get* RA=F39E.

**Note: These two have been exchanged in order.




4.2. Convert the result in RA to decimal equivalent using two's complement convention and state what the sequence of instructions does based on the contents of RA and the initial value.

**Answer** RA = -3170. The sequence .... *Multiplies by 10*

Notice that the two's complement interpretation of F39E is -3170, and we must keep the same convention used at the beginning. *We cannot change convention during an application unless it is part of the program planning.*

4.3. Based on your answer to 4.2, add a comment to each instruction so that it makes sense with the objective of the sequence.

**Answer** *Wording may be different but the message should be similar*

1: $RA \leftarrow -317$ *; A = X (-317 = FEC3h)*
2: $RA \leftarrow RA + RA$ *; now A = 2X*
3: Push RA ; *Save 2X*
4: $RA \leftarrow RA + RA$ *; A = 2X + 2X = 4X –multiply by 4*
5: $RA \leftarrow RA + RA$ *; A = 4X + 4X = 8X –multiply by 8*
6: Pop RB *; retrieve 2X*
7: $RA \leftarrow RA + RB$ *; A = 8X + 2X = 10X -multiply by 10*

4.4. If we take the contents of RA after the first instruction in hex equivalent, and convert the final result to decimal using normal unsigned binary convention, can the sequence be interpreted as in 4.2 above? Justify your answer.

**Answer** (Yes or NO) NO.

Justification: The equivalent of 0xFEC3 is 65,219. Even a multiplication by 2 yields a number requiring more than 16-bits. Hence, the algorithm would apply only if the length of the registers were at least 20 bits.

(REMARK:)
As always, besides the algorithm we should consider the range of applications. For 16 bits, unsigned numbers are in the range [0, 65535], which means that the largest number that can be multiplied by 10 in this problem is 6,553. 16-bit signed numbers fall in the interval [-32768, 32767] so our valid interval of numbers to multiply by 10 in this problem would be [-3276, 3276].

**Problem 5**. We have two pieces of code: The caller (main program if you will) and the callee, a subroutine.

The caller executes three (3) different push operations before calling the subroutine. The address of the next instruction in sequence after the call is 0xFA00. Assume that SP = 0x025A just before the subroutine call, and that SP points to the address of the last item pushed onto the stack.

Once inside the subroutine there are five (5) different pushes that are executed along with six (6) different pop instructions before the return operation is carried out.

The programmer notices that the program did not execute properly.

5.1. Draw a picture of the stack illustrating what was explained above. You have to indicate the exact addresses of all the memory locations used in the stack during the run of the caller and callee. The contents of the memory location that was used and is unknown to you should be filled with YYYY. If the memory location was not used and the contents is unknown, write XXXX. If the memory contents is known, then you have to write it down. Show clearly where SP is pointing at the end of the sequence.

5.2. Explain what you believe is the reason for the program malfunction.

**ANSWER**

*Because the address in PC is not of the instruction following the call. Hence, the instruction to be executed is not the intended one. It may even be an invalid address or and invalide code*

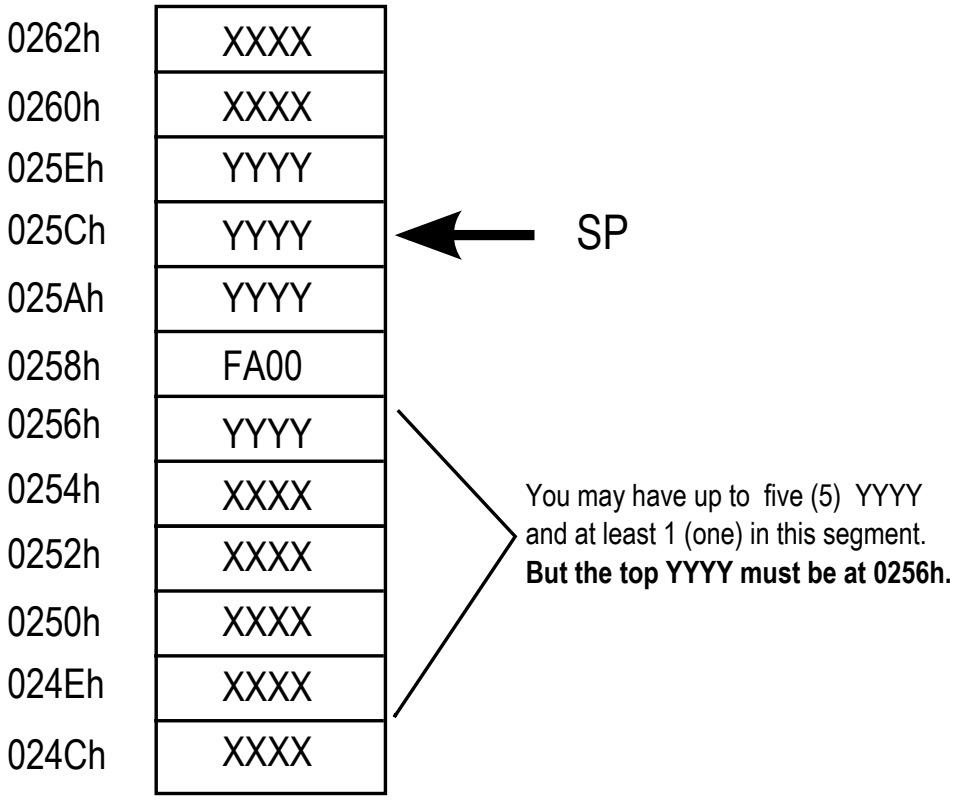
5.3. (BONUS) How would the MSP430 have reacted if this would have happened using the Launchpad?

**ANSWER**

*If the PC is loaded with an invalid address or the instruction to be executed is invalid, the system resets.*

..... **ANSWER to 5.1** .....

**ANSWER**: *Just before the call, assume SP= 025Ah is pointing at the third item pushed at that moment. Hence, addresses 025Ah, 025Ch and 025Eh should be filled with YYYY.*
*The address FA00h is pushed onto address 0258h.*
*There is no indication as how the sequence of five pushes and six pops was done. We can assume that a pop followed a push, for example (any other valid consideration is OK). With this assumption, the pushed item went to memory location h, and the pop brought back SP to 0258h. Since there was an extra pop before returning, at the moment of return the stack pointer points at SP = 025Ah, which is not the place where FA00h was stored, but where the last item before the last call is stored. This value is popped to PC on return, and SP points at 025Ch*


0262h
XXXX
0260h
XXXX
025Eh
YYYY
025Ch
YYYY
SP
025Ah
YYYY
0258h
FA00
0256h
YYYY
0254h
XXXX
0252h
XXXX
0250h
XXXX
024Eh
XXXX
024Ch
XXXX
You may have up to five (5) YYYY
and at least 1 (one) in this segment.
**But the top YYYY must be at 0256h.**