Answers to the September 9, 2016, test of Pearls of Computer Science

P.T. de Boer, UT/EWI/DACS, 2016-10-24

---

1. Binary numbers

(a) Two-complement notation is a position system, so from left to right the digits have weights $-2^5, +2^4, ..., 2^0$, so in decimal this number is $-32 + 4 = -28$.

(b) It's 64 in hexadecimal, since $6 \times 16^1 + 4 \times 16^0 = 100$.

(c) Shift left by one position. This works because it moves every bit to a position whose weight is double that of its original position. The free position at the right should be set to 0, otherwise it would contribute to the value.

There are two possible interpretations of this problem (it was not stated clearly enough), fortunately with the same conclusion:
• after shifting, it becomes a 9-bit number, of which the left-most bit is what was the left-most bit of the original 8-bit number, so the sign is preserved and nothing special is needed.
• after shifting, the left-most bit drops out of the register so an 8-bit number remains (this is what happens if you do this on a real 8-bit processor, like the Arduino); even in that case, no special treatment is needed, if the original number was small enough to still fit in 8 bits after doubling (i.e., it was between –64 and +63): in that case, in the original number the 2 left-most bits were equal, so also after shifting and dropping 1 one of them, the remaining left-most bit still represents the sign correctly.

(d) If the left-most bit was 0, the error is also 0.
If the left-most bit was 1, then you counted it as 2^{n-1}, while it should be counted as -2^{n-1}, so the error is 2^n.

---

2. Boolean logic

(a)

A	B	C	D
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(b) It's a four-input OR gate. You can show this in different ways, e.g. by writing down the entire truth table, or by applying some Boolean algebra: output = $\overline{(\overline{A+B}) \cdot (\overline{C+D})}) = (A+B)+(C+D) = A+B+C+D$.

(c) Respectively: distributive, complement, identity, distributive, complement, identity.

(d) By Boolean algebra, we can write $A + (B \cdot \overline{C}) = \overline{\overline{A} \cdot \overline{(B \cdot \overline{C})}}$ so the circuit consists of using two NAND gates to invert A and C, feeding $\overline{C}$ and B to a NAND gate, and feeding its output and $\overline{A}$ to another NAND gate which gives the final output.
(On the exam you should really draw the schematic instead of the above textual description, but this is easier to quickly type for me...)

3. Problem 3

```
            read addr 1 /
            write addr    read addr 2   instruction
Time slot 0    2             3            0
Time slot 1    1             2            1
Time slot 2    4             1            0
```

Note that the last step only serves to copy the result to R4, but since this (rather weird) processor does not have a copy or move instruction, the only way is to use OR, knowing that R4 is already 0.

4. Problem 4

Write down step by step how the registers change (all hex); if nothing is written, it means the number stays unchanged:

```
  R0  R17   R18   R19   R20   R21
      3
            2
                  1
                        0
  6
                        6
                        5
      2
                              2
                              1
                                    do jump, back to the mul
  4
                        9
                        8
      1
                              1
                              0
                                    do not jump
```

So R20 contains now 0. Total 18 steps, each of which takes 1 clock cycle, except 1 step, namely the BRNE that indeed jumps, which takes 2 clock cycles. Thus 19 clock cycles in total.

There was some confusion about the description saying that the MULtiply instruction places the “least-significant part” of its result in R0. This is what the real AVR / Arduino does, the 8 least-significant bits go to R0, and the 8 most-significant bits go to R1 (since the outcome of an 8 bit times 8 bit multiply can be up to 16 bits long). In this, the mnemonic deviates from the usual convention on AVR processors that the result is written into the register mentioned before the comma. We’ll try to prevent such confusion in future exams...

5. Problem 5

It calculates $Y \cdot X + X \cdot Y = 2XY$.