# Makeup Quiz—30 Nov 2009

*CS 353—Architecture and Compilers—Fritz Ruehr*

Name: ______________________

1. Say that a certain computer has a word size of **18** bits, and that there are **32** operations encoded *uniformly* using the first several bits of an instruction word. This machine has a `PUT` instruction that allows the programmer to write an *immediately specified* data value into a given address of RAM, as follows:

   `PUT` *data, address*

   If there are **512** different RAM locations addressable by this instruction, how many bits will be left to specify the data to be stored?

   ______________

2. Assume that the regular expressions R and S represent sets of strings with 3 and 5 members, respectively. Put a *lower bound* (how few are possible?) and an *upper bound* (how many) on the number of strings in each of the following regular expressions:

|  | *lower bound* | *upper bound* |
|---|---|---|
| • R \| S | ______________ | ______________ |
| • R · (a \| b) | ______________ | ______________ |
| • R* | ______________ | ______________ |

3. Say that we have a code template representing some mathematical operation of interest, perhaps multiplication of the values held in two registers. The idea is that when we plug in this code, we specify a couple of registers like this:

```
...              ; R7 and R4 have some values here
MULT R7, R4      ; expands into code to multiply R7 by R4
...              ; product in R4, everything else unaffected
```

   and the code will be inserted so that the values that were in `R7` and `R4` from the previous line get multiplied together, with the result left in `R4`, say. Of course, some *other* registers would get **used by the code** to do this, say **`R8`**, **`R9`**, **`J0`** and the data register, **`DR`**, to be specific. In order to make sure that the `MULT` code works, it will begin by *saving the values* from a number of registers into RAM, and end by *restoring those values* back from RAM.

   Circle the registers that should be *explicitly saved and restored* by the `MULT` code:

   `R0   R1   R2   R3   R4   R5   R6   R7   R8   R9   J0   J1   J2   J3   DR   PC`

4. Could the following sequences of bytes represent a valid UTF-8 encoding of Unicode data? For each "**NO**" answer, *circle the first offending byte.*

- `01110110 10101110 01101111 11010111 10110000` ______________
- `01100011 11010011 11001101 10111010 01111011` ______________
- `11011111 10000001 01100101 01111101 01010101` ______________