Computation Structures — Tutorial 1

September 15, 2015

Contact Information

- Practice session by David Taralla
- dtaralla@ulg.ac.be, Office 2.96 (B28).
- http://www.montefiore.ulg.ac.be/~dtaralla/course_cs/

Microcode for ULG01

1. Give symbolic `ULg01` microcode for instruction `SWAPIFZ(Ra,Rb,Rc)`. When contents of register `Rc` is 0, this instruction swaps contents of `Ra` and `Rb` registers. Otherwise, it has no effect.

2. Combine `LD(Ra,Lit,Rd)` and `JMP(Rd,Rc)` in a single instruction. This new instruction `JMPI(Ra, Lit, Rc)` directs the program to an address found in memory at the address `Ra + Lit`. Register `Rc` shall receive the address of the instruction immediately following the JMPI we're executing. Provide the symbolic `Ulg01` microcode for `JMPI(Ra, Lit, Rc)`

3. Give symbolic `Ulg01` microcode for the following instruction:

```
BUZ(Ra,Lit,Rc): PC <- PC + 4
                EA <- PC + 4 * SEXT(Lit)
                TMP <- Reg[Ra]
                Reg[Rc] <- PC
                if TMP < 0 then PC <- EA
```

   This instruction saves the program counter's value in register `Rc` and branches `Lit` instructions away iff the contents of register `Ra` is negative (**U**nder **Z**ero).

4. Give symbolic `Ulg01` microcode for the following instruction:

```
JMPODD(Ra, Rb, Rc): PC <- PC + 4
                    TMP <- Reg[Rb] % 2
                    EA <- Reg[Ra] & 0xFFFFFFFC
                    Reg[Rc] <- PC
                    if TMP = 1 then PC <- EA
```

   Where `a % b` is the modulo operator, computing the remainder of integer division of a by b. Note that the ALU has no support for `A*B, A/B` nor `A%B`