Computation Structures — Tutorial 4

October 25, 2016

μ-code for ULg03 – Virtual Memory

Reminder

- ULg03 introduces *virtual memory*. In practice, it translates into a more complex paradigm to access data and code stored in DRAM.
- With virtual memory, user programs think they have access to a large contiguous space of dynamic memory, while physically it is fragmented and partially on disk.
- Purpose of virtual memory: keep memory representation simple for user programs without impacting the performances. Indeed, reserving a fixed contiguous memory space for each process is excessively constraining for the system and the processes themselves.
- Summary of changes made to ULg02 to implement VM:
  1. New DRAM module (DMAR has been replaced by a new circuit).
  2. New control signals output by the control ROM:
     - LD and DR signals for UCVP, UCPP, UDVP, UDPP;
     - UC/D: 0 for code and 1 for data.
  3. Constants ROM updated with the addresses to two new handlers (*cache miss code* and *cache miss data*).
  4. Micro-code can be longer (up to 32 phases).

Exercises

1. 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 ULg03 user *and* supervisor micro-code for the JMPI(Ra, Lit, Rc) instruction.
2. (January 2017) In order to simplify the microcode of ULg03, one of your colleagues proposes to implement a hardware comparison of cached virtual page addresses. Practically, the idea consists in comparing the page numbers stored in the TLBs (UCVP and UDVP) with a value on the bus and using the result of the comparisons as input of the control ROM.

(a) Complete the wiring of the given virtual memory and control unit diagrams (see Figure 1) to implement the comparison mechanism and give relevant names to the various signals you need for controlling and using it[1].

(b) Using the signals you have defined, implement the microcode of the new instruction `LDGT(Ra, Rb, Rc)` in user-mode. This instruction loads the value stored at the address found in `Ra` into `Rc` if this value is greater than `Rb`, otherwise it loads 0 in `Rc`.

```
LDGT(Ra, Rb, Rc): PC <- PC + 4
                  TMP <- Mem[Reg[Ra]]
                  if TMP > Reg[Rb] then
                      Reg[Rc] <- TMP
                  else
                      Reg[Rc] <- 0
```

3. We wish to upgrade `ULg03` DRAM module so that the machine memory is 8MB instead of 4MB. There are two ways of doing that; indicate them on the provided schemas (see Figure 2). Explain briefly.

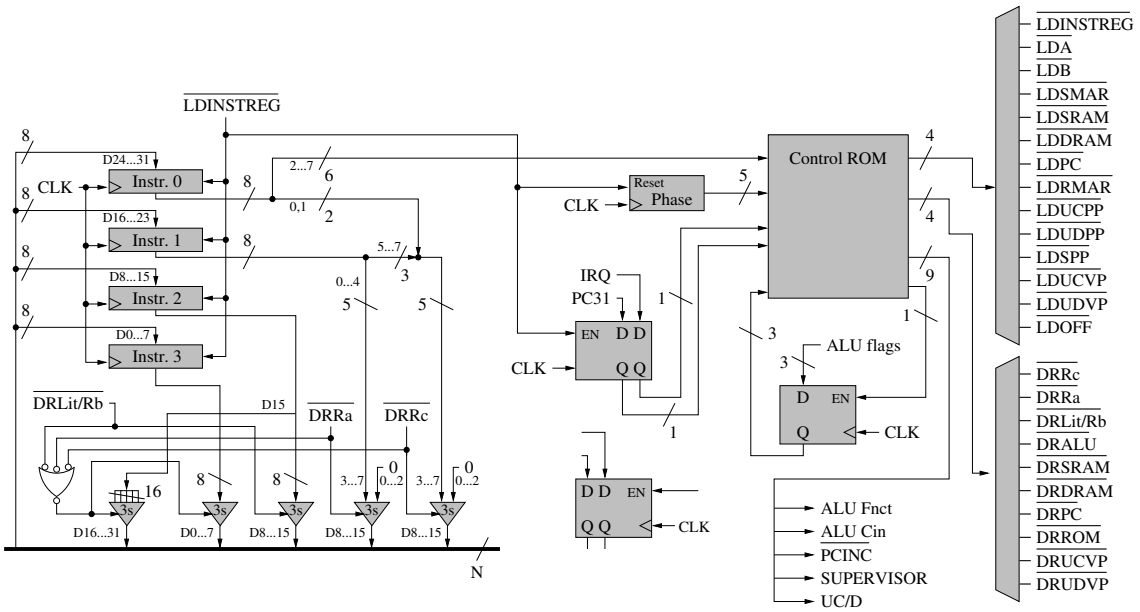
Supplementary exercises

1. Provide the `ULg03` user *and* supervisor micro-code for the `ADDI(Ra, Rb, Rc)` instruction. It adds the value of `Rb` with the value at address `Ra` in the dynamic memory and stores the result in `Rc`.

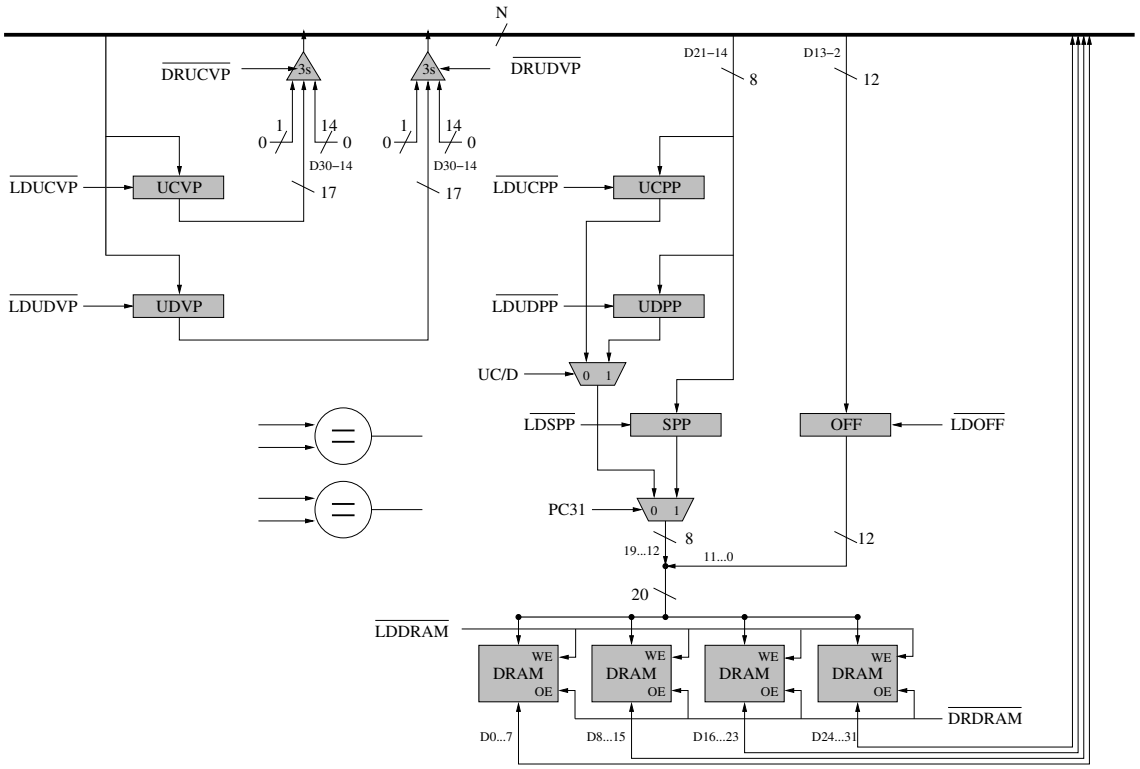
```
ADDI(Ra, Rb, Rc): PC <- PC + 4
                  Reg[Rc] <- Mem[Reg[Ra]] + Reg[Rb]
```

2. Provide the `ULg03` user *and* supervisor micro-code for the `BNE(Ra, label, Rc)` instruction.

[1] You can define new signals if needed, including microcode ROM outputs.



(a) Control unit (*hint*: notice the new unconnected memory component)



(b) Virtual RAM (*hint*: notice the new unconnected comparators and the new output wiring of UCVP and UDVP, which are no longer 3 state)

Figure 1: ULg03 with hardware-comparison of virtual addresses

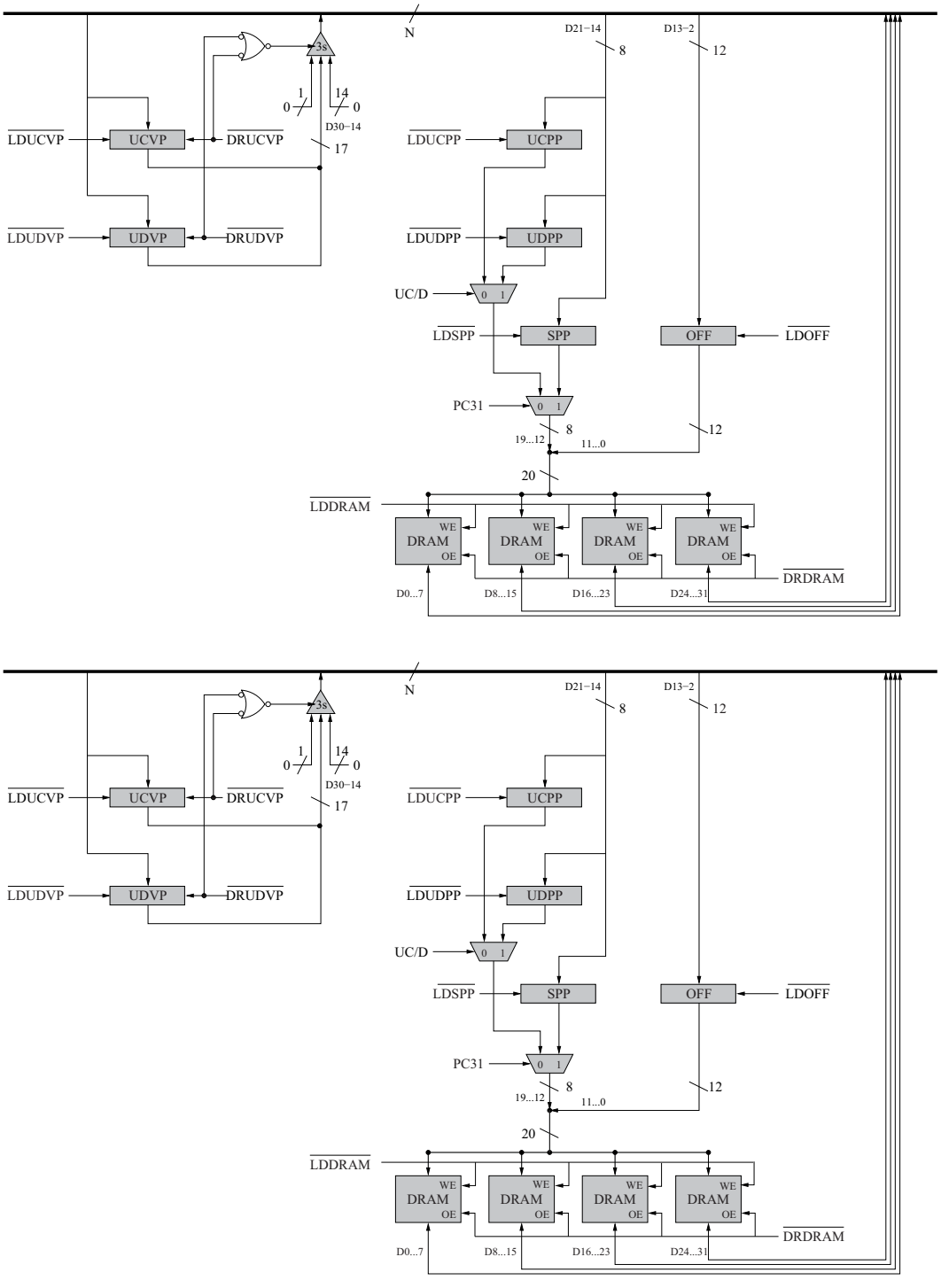


Figure 2: Dynamic memory circuit in ULg03