Computation Structures — Tutorial 4 Erratum

October 28, 2016

1 Exercice 2a (`JMPI` in user mode)

A small mistake has been made when writing the *Cache miss data* branch of the user μ-code at exercise 2 (see Table 1).

Ph.	Flags	ALU	DR	LD	
11	E = 0	0xffffffff	ALU	RMAR	
12			ROM	SMAR	
13			PC	A	
14		A - 1	ALU	SRAM	
15		0xffffffff	ALU	A	
16		A-1	ALU	RMAR	
17			ROM	PC	SVR
18			PC	OFF	
19			PC	SPP	PC+
20			DRAM	INSTREG	

Table 1: Cache miss data branch, as defined during the exercise session.

Indeed, this μ-code is not compatible with the handler you have seen during the lecture. Especially, the *Cache miss data* handler is the following:

```
h_stub:
   SUBC(XP, 4, XP)      | plan to reexecute the instruction
                        | that has been suspended
   ST(r0, User, r31)    | save
   ST(r1, User+4, r31)
   . . .
   ST(r30, User+30*4)
   CMOVE(KStack, SP)    | Load the system SP
   RDUDVP(r1)           | the virtual page address to be
                        | translated
   PUSH(r1)             | pass it as argument
   BR(CMHandler,LP)     | call the Handler
   DEALLOCATE(1)        | remove the argument from the stack
   WRUDPP(r0)           | install the returned value
   LD(r31, User, r0)    | restore
   LD(r31, User+4, r1)
   LD(r31, User+30*4, r30)
   JMP(XP)              | return to application
```

I guess that you have noticed that the first instruction of this handler consists in decrementing XP by 4. Indeed, because of the cache miss data, we weren't able to execute the instruction that triggered the cache miss. Therefore, now that the User Data Physical Page (UDPP) register contains a valid page number, we have to re-execute this instruction (of which the address is effectively Reg[XP] - 4). The error in the μ-code in Table 1 lies at the phases 13 and 14 because we also decrement the value of PC before storing it into Reg[XP]. Therefore when the handler is executed, Reg[XP] already contains the address of the instruction we want to jump back to and this address is decremented by 4. Therefore, we don't go back to the instruction that triggered the cache miss but the one before.

To be consistent with the handler seen during the lectures, the μ-code should not store in XP the address of the current instruction but the address of the next. This is shown in the corrected μ-code in Table 2. Note that in the new μ-code, we use one phase less than before for this branch so this impacts the rest of the μ-code (especially, the number of no-op operations to be performed for the *cache hit* branch and the phases for the subsequent operations).

At the exam, you are expected to be consistent with the conventions seen during the lectures.

Ph.	Flags	ALU	DR	LD	
11	E = 0	0xffffffff	ALU	A	
12		A	ALU	RMAR	
13			ROM	SMAR	
14			PC	SRAM	
15		A-1	ALU	RMAR	
16			ROM	PC	SVR
17			PC	OFF	
18			PC	SPP	PC+
19			DRAM	INSTREG	

Table 2: Cache miss data branch, corrected and consistent with the handler seen during the lecture.

2 Exercice 2b (JMPI in SVR mode)

A second small mistake was made in the supervisor mode version of JMPI. Indeed, in this case, we have to check whether we have to switch to user mode or remain in supervisor mode. This check is done by looking at the PC31 bit of the address **we want to jump to**.

The μ-code given at the tutorial actually checks the PC31 bit of the address Reg[Ra] + Lit which is **not** the address we want to jump to. This μ-code is given in Table 3.

To correct this mistake, one has to check the PC31 bit of the address found in Mem[Reg[Ra] + Lit] (which is the address we want to jump to). See corrected version in Table 4.

Ph.	Flags	ALU	DR	LD	
5	...	...	...	...	
6		A+B	ALU	OFF	
7		A+B	ALU	A	LF
8	N = 1		DRAM	PC	SVR
9			PC	OFF	
10	...	...	...	...	

Table 3: μ-code as given during the tutorial. The result of `A+B` is actually `Reg[Ra] + Lit` and not the address we want to jump to.

Ph.	Flags	ALU	DR	LD	
5	...	...	...	...	
6		A+B	ALU	OFF	
7			DRAM	A	
8		A	ALU	A	LF
9	N = 1		DRAM	PC	SVR
10			PC	OFF	
11	...	...	...	...	

Table 4: Corrected μ-code: now, one first fetches the address from `Mem[Reg[Ra] + Lit]` before checking the PC31 bit.