Saarland University
Department 6.2 – Computer Science
Dr. M. A. Hillebrand
Prof. Dr. W. J. Paul

Computer Architecture I – WS 06/07

Exercise Sheet 5 **(due: 27.11.06)**

Note: For the completion of this exercise sheet, make use the schematics and the control automaton, which are fully specified in the current version of the lectures notes. In particular note, that the definition of the predicates $alu(I)$ and $alui(I)$ has been strengthened.

Exercise 1: (ISA; predicates and instructions) **(4+3+3+4+4=18 points)**

1. In class the predicate $illegal(I)$ was mentioned. It indicates whether an instruction $I \in \mathbb{B}^{32}$ is not a legal instruction. Specify the disjunctive normal form (DNF) of this predicate.
2. What would the machine do on input of the following three instructions in case illegal instructions would not be handled specially:
   - 000000 00000 00000 00000 000000000 01
   - 100111 00110 01010 01100 000001100 11
   - 111010 01000 00010 00000 000000011 11
3. For the processor presented in class, we had the assumption that memory requests (for fetch as well as for load / store instructions) have to be aligned, i. e., the memory access width $d(I)$ has to divide the delayed PC $\langle dpc \rangle$ or the effective address $\langle ea(I) \rangle$, respectively.

   What does the machine currently do if this assumption is violated, i.e., say, if there is a misaligned load / store access?
4. Construct a circuit that checks whether a load / store request is *not* aligned, signalling this condition with an output signal $mal \in \mathbb{B}$.
5. Let the predicate $nop(I[31:0])$ be defined as:

$$\forall c.\ (c.m_4(c.DPC) = I) \Rightarrow$$

$$(c'.gpr = c.gpr) \wedge (c'.m = c.m) \wedge (c'.DPC = c.PCP) \wedge (c'.PCP = c.PCP +_{32} 0^{29}100)$$

   An instruction $I(c)$ is a NOP-instruction (no operation) if $nop(I(c))$ holds. That means, a NOP-instruction is an instruction with the only effect that the next sequential instruction after the PCP will be executed in the next step.

   List all legal NOP instructions.

Exercise 2: (ISA; PC calculation and program correctness) (2+4+5=11 points)

Given the following memory content: (The value before the colon denotes the memory address of the instruction; the blanks are just for readability)

```
 0 : 000010 11111 11111  11111 111111111 00
 4 : 010110 00010 00000  00000 000000000 00
 8 : 010110 00000 00000  00000 000000000 00
12 : 001011 00010 00010  00000 000000001 00
16 : 010110 00000 00000  00000 000000000 00
20 : 001011 00010 00010  00000 000000001 00
24 : 010110 00000 00000  00000 000000000 00
28 : 001011 00010 00010  00000 000000001 00
```

Assume the initial value of GPR(2) is 28.

1. Write the program in a readable form with the help of the instruction names defined in class (cf. Exercise 3).
2. Perform 15 execution steps and specify in each step the value of $c.PCP$, $c.DPC$, and $c.gpr(2)$.
3. Prove the correctness of the first 7 executions by applying the formal semantics from the lecture in each step to your current configuration.

Exercise 3: (ISA; programming) (6 points)

Give a ISA program that counts the number of leading zeros i of a bit string $0^i 1 \star^{32-i-1}$ stored in $c.gpr(0^4 1)$ (this number may be 32 for $c.gpr(0^4 1) = 0^{32}$). Each line of your 'source code' should have the following form:

'address of instr.': 'instruction' 'register addresses or constants' // 'comments'

for example:

0: addi RS1=1 RD=5 imm=32 // gpr(5) = gpr(1) + 32 because . . .

Your program should start at memory address 0 and should write the final result i to memory address 400 (in case you need more than 100 lines for your program, write the result to address 4000). Comment each line of your code! Otherwise you will earn 0 points for this exercise.