Saarland University
Department 6.2 - Computer Science
Prof. Dr. W. J. Paul
M. Sc. Christian Müller

Computer Architecture I - WS 07/08

Exercise Sheet 6

Outline

Last week we have finished the DLX specification and started with a hardware implementation of our processor. The hardware we want to build (DLX implementation), should cover the complete computation described by the DLX specification. In the following exercises we will implement some of needed hardware parts.

Excercise 1: (GPR implementation)

In this exercise we will construct the hardware counterpart of the set of registers GPR from the DLX specification. That is, we need a circuit that fulfills the following properties:

1. The GPR should be based on a 3-Port RAM, i.e. your construction should have inputs $w \in \mathbb{B}, Din, Aad, Bad$, and Cad as well as outputs DoA and DoB.
2. The address width should be 5, i. e. the three addresses Aad, Bad, Cad are in $\mathbb{B}^5$.
3. The data width is 32-bit, i.e. Din, DoA, and DoB are in $\mathbb{B}^{32}$.
4. At read address 0 the data word 0^{32} is returned at the output and the address 0 cannot be written.[1]

Formally we should have:

$$DoA^i = \begin{cases} 0^{32} & : Aad^i = 0^5 \\ gpr^i(Aad) & : Aad^i \neq Cad^i \vee \neg w^i \\ \text{unspecified} & : \text{otherwise} \end{cases}$$

$$DoB^i = \begin{cases} 0^{32} & : Bad^i = 0^5 \\ gpr^i(Bad) & : Bad^i \neq Cad^i \vee \neg w^i \\ \text{unspecified} & : \text{otherwise} \end{cases}$$

$$gpr^{i+1}(a) = \begin{cases} Din^i & : Cad^i = a \wedge a \neq 0 \wedge w^i \\ gpr^i(a) & : \text{otherwise} \end{cases}$$

Excercise 2: (constant computation)

In the specification we sometimes use a constant as the right operand in arithmetic and shift operations. This constant was specified in the class as $C_0(c)$ and contains, according to the currently executed instruction, either the sign extended immediate constant or the shift amount. Implement hardware, which provides us this constant – $C_0(h)$.

[1] This was **not** mentioned in the class!

Saarland University
Department 6.2 - Computer Science
Prof. Dr. W. J. Paul
M. Sc. Christian Müller

Computer Architecture I - WS 07/08

Exercise Sheet 6

Excercise 3: (DLX implementation)
Our instruction set allows to store in the memory not only whole words (bitvectors of size 32), but also halfwords (16 bits) and even single bytes (8 bits). The value to store is always taken from $GPR(a)[8 \cdot d(h) - 1 : 0]$ for the access width $d(h) \in \{1, 2, 4\}$. Moreover, we use the two least significant bits of $ea(h)$ in order to determine the position of the stored halfword or byte. For example, if we use the instruction *sh* and $ea(h)[1:0] = 00$, then the first halfword will be written to the word address $ea(h)[31:2]00$; for $ea(h)[1:0] = 10$ the second halfword will be written.

1. Estimate the minimal delay of the $ea(h)$ computation.
2. Show, that we can use the bits $ea(h)[1:0]$ before the entire effective address is computed.

Excercise 4: (ISA programming)
Give an assembler program that counts the number of leading zeros i of a bit string $0^i 1^{32-i}$ stored in $c.gpr(0^4 1)$. 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).