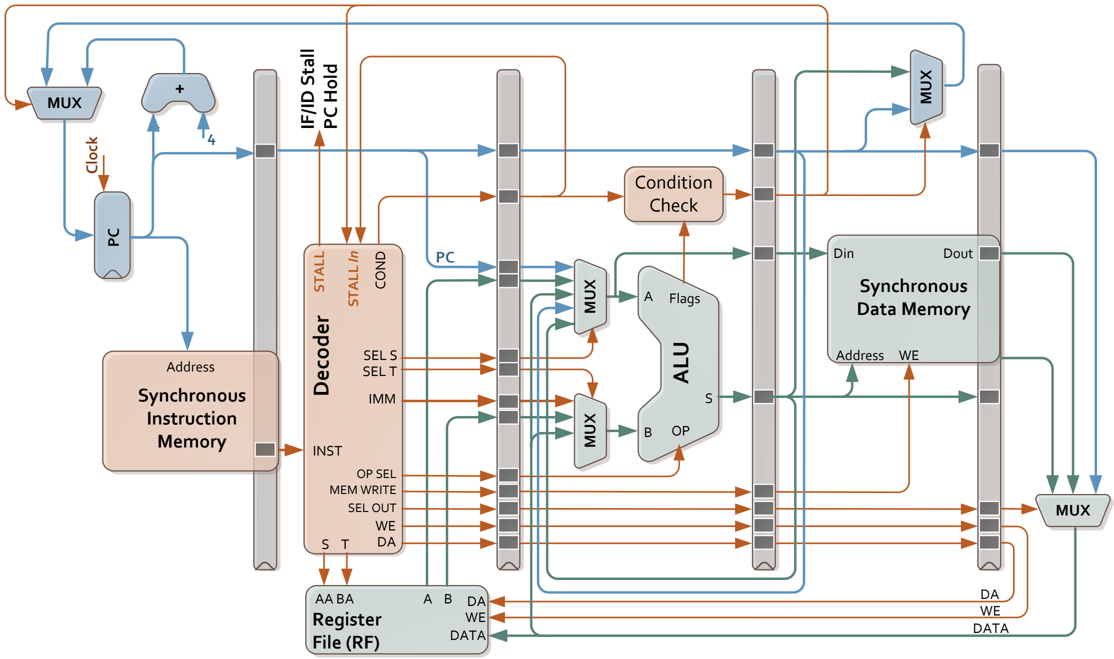
Advanced Computer Architectures

2015/2016



Lab I

Development of a simple RISC processor

Lab demonstration: March 29

Report due on April 3

Department of Electrical and Computer Engineering

Instituto Superior Técnico

February 2016

1. Introduction

The objective of the first lab is the development of a simple processor supporting the MicroBlaze Instruction Set Architecture, which has the following characteristics:

- Instruction and data words of 32 bits;
- 32 general purpose registers of 32 bits (`r0` to `r31`), with the value of register `r0` being fixed at zero (all writes to register `r0` are discarded).
- 1 Special Purpose Register (`rIMM`) of 16 bits for immediate (constant) storage (see the `IMM` instruction);
- 1 *Machine Status Register* (`MSR`), holding the carry flag (C) and the immediate flag (I) fields;
- a 32-bit memory address space, organized at the byte level, with all reads and writes being made in Little Endian format;
- No support for miss-aligned memory accesses.

MicroBlaze Instruction Set Architecture (ISA)

The MicroBlaze Instruction set includes arithmetic, logic, shift and comparison operations over integer and single-precision floating point numbers. Additionally, it supports operations on peripherals, such as those connected through the Xilinx Fast Simple Link (FSL), on instruction and data caches and on the Translation Look-Aside Buffer (TLB). However, only a small but representative subset of instructions are required for most programs. Hence, for the scope of the lab, only arithmetic, logic, shift and comparison operations over integer operands, as well as load/store and control operations need to be supported.

To facilitate the description of the MicroBlaze Instruction set architecture, the following naming convention will be used:

U(RA), U(RB) – Operands RA and RB should be considered as *unsigned* (by default all operands are represented in 2's complement).

C – Carry bit, which value is stored on the *Machine Status Register (MSR*) whenever the following instructions are executed: ADD,RSUB,ADDI,RSUBI,ADDC,RSUBC,ADDIC,RSUBIC,SRA,SRC,SRL.

Rx{a:b} – Bits *a* and *b* of register Rx; as an example, R3{7:0} represents the 8 least significant bits of register R3.

S(IMM) – Sign extension of the 16-bit immediate value IMM. If MSR[I]=0 the extension should be performed using 2's complement sign extension, otherwise extension should be performed using the value store in register rIMM.

M[A + B] – Indexing of memory position with address A + B.

The full set of instructions to be supported is described in Tables 1, 2, 3 and 4. Although not explicitly stated in the tables, with the exception of the IMM instruction, all other instructions should also set the MSR[I] flag to zero, in order to guarantee that the extension of a 16-bit immediate value IMM_{16} is performed as follows:

IMM_{32} $\leftarrow$ rIMM{15:0},IMM_{16}{15:0}, if the current instruction is preceded by an IMM instruction
$\leftarrow$ sign extension(IMM_{16}{15:0}), otherwise

where IMM_{32} is the resulting 32 bit immediate value.

Table 1 – Arithmetic, logic and shift instructions.

Assembly mnemonic and operands		Instruction word					Operation
		31-26	25-21	20-16	15-11	10–0	
ADD	Rd,Ra,Rb	000000	Rd	Ra	Rb	000 0000 0000	$RD \leftarrow RB + RA$
RSUB	Rd,Ra,Rb	000001	Rd	Ra	Rb	000 0000 0000	$RD \leftarrow RB + \overline{RA} + 1$
ADDC	Rd,Ra,Rb	000010	Rd	Ra	Rb	000 0000 0000	$RD \leftarrow RB + RA + C$
RSUBC	Rd,Ra,Rb	000011	Rd	Ra	Rb	000 0000 0000	$RD \leftarrow RB + \overline{RA} + C$
ADDK	Rd,Ra,Rb	000100	Rd	Ra	Rb	000 0000 0000	$RD \leftarrow RB + RA$
RSUBK	Rd,Ra,Rb	000101	Rd	Ra	Rb	000 0000 0000	$RD \leftarrow RB + \overline{RA} + 1$
ADDKC	Rd,Ra,Rb	000110	Rd	Ra	Rb	000 0000 0000	$RD \leftarrow RB + RA + C$
RSUBKC	Rd,Ra,Rb	000111	Rd	Ra	Rb	000 0000 0000	$RD \leftarrow RB + \overline{RA} + C$
CMP	Rd,Ra,Rb	000101	Rd	Ra	Rb	000 0000 0001	$RD \leftarrow RB + \overline{RA} + 1$ $RD[31] \leftarrow 0$, if $RB \geq RA$ $RD[31] \leftarrow 1$, else
CMPU	Rd,Ra,Rb	000101	Rd	Ra	Rb	000 0000 0011	$RD \leftarrow RB + \overline{RA} + 1$ $RD[31] \leftarrow 0$, if $U(RB) \geq U(RA)$ $RD[31] \leftarrow 1$, else
ADDI	Rd,Ra,Rb	001000	Rd	Ra	Imm		$RD \leftarrow S(Imm) + RA$
RSUBI	Rd,Ra,Rb	001001	Rd	Ra	Imm		$RD \leftarrow S(Imm) + \overline{RA} + 1$
ADDIC	Rd,Ra,Rb	001010	Rd	Ra	Imm		$RD \leftarrow S(Imm) + RA + C$
RSUBIC	Rd,Ra,Rb	001011	Rd	Ra	Imm		$RD \leftarrow S(Imm) + \overline{RA} + C$
ADDIK	Rd,Ra,Rb	001100	Rd	Ra	Imm		$RD \leftarrow S(Imm) + RA$
RSUBIK	Rd,Ra,Rb	001101	Rd	Ra	Imm		$RD \leftarrow S(Imm) + \overline{RA} + 1$
ADDIKC	Rd,Ra,Rb	001110	Rd	Ra	Imm		$RD \leftarrow S(Imm) + RA + C$
RSUBIKC	Rd,Ra,Rb	001111	Rd	Ra	Imm		$RD \leftarrow S(Imm) + \overline{RA} + C$
MUL	Rd,Ra,Rb	010000	Rd	Ra	Rb	000 0000 0000	$RD \leftarrow RA \times RB$
MULH	Rd,Ra,Rb	010000	Rd	Ra	Rb	000 0000 0001	$RD \leftarrow RA \times RB \gg 32$
MULHU	Rd,Ra,Rb	010000	Rd	Ra	Rb	000 0000 0011	$RD \leftarrow U(RA) \times U(RB) \gg 32$
MULHSU	Rd,Ra,Rb	010000	Rd	Ra	Rb	000 0000 0010	$RD \leftarrow RA \times U(RB) \gg 32$
BSRA	Rd,Ra,Rb	010001	Rd	Ra	Rb	010 0000 0000	$RD \leftarrow RA \gg U(RB[4{:}0])$ (arithmetic shift)
BSLA	Rd,Ra,Rb	010001	Rd	Ra	Rb	110 0000 0000	$RD \leftarrow RA \ll U(RB[4{:}0])$ (arithmetic shift)
BSRL	Rd,Ra,Rb	010001	Rd	Ra	Rb	000 0000 0000	$RD \leftarrow RA \gg U(RB[4{:}0])$ (logic shift)
BSLL	Rd,Ra,Rb	010001	Rd	Ra	Rb	100 0000 0000	$RD \leftarrow RA \ll U(RB[4{:}0])$ (logic shift)
BSRAI	Rd,Ra,Rb	011001	Rd	Ra	0000	010 000,Imm5	$RD \leftarrow RA \gg U(Imm5)$ (arithmetic shift)
BSLAI	Rd,Ra,Rb	011001	Rd	Ra	0000	110 000,Imm5	$RD \leftarrow RA \ll U(Imm5)$ (arithmetic shift)
BSRLI	Rd,Ra,Rb	011001	Rd	Ra	0000	000 000,Imm5	$RD \leftarrow RA \gg U(Imm5)$ (logic shift)
BSLLI	Rd,Ra,Rb	011001	Rd	Ra	0000	100 000,Imm5	$RD \leftarrow RA \ll U(Imm5)$ (logic shift)
OR	Rd,Ra,Rb	100000	Rd	Ra	Rb	000 0000 0000	$RD \leftarrow RA\ or\ RB$
AND	Rd,Ra,Rb	100001	Rd	Ra	Rb	000 0000 0000	$RD \leftarrow RA\ and\ RB$
XOR	Rd,Ra,Rb	100010	Rd	Ra	Rb	000 0000 0000	$RD \leftarrow RA\ xor\ RB$
ANDN	Rd,Ra,Rb	100011	Rd	Ra	Rb	000 0000 0000	$RD \leftarrow RA\ and\ \overline{RB}$
SRA	Rd,Ra	100100	Rd	Ra	0000	000 0000 0001	$(RD, MSR[C]) \leftarrow (RA[31], RA)$ (arithmetic shift right with carry)
SRC	Rd,Ra	100100	Rd	Ra	0000	000 0010 0001	$(RD, MSR[C]) \leftarrow (MSR[C], RA)$ (rotate right with carry)
SRL	Rd,Ra	100100	Rd	Ra	0000	000 0100 0001	$(RD, MSR[C]) \leftarrow (0, RA)$ (logic shift right with carry)
SEXT8		100100	Rd	Ra	0000	000 0110 0000	$RD \leftarrow S(RA[7{:}0])$
SEXT16		100100	Rd	Ra	0000	000 0110 0001	$RD \leftarrow S(RA[15{:}0])$
ORI	Rd,Ra,Imm	101000	Rd	Ra	Imm		$RD \leftarrow RA\ or\ S(Imm)$
ANDI	Rd,Ra,Imm	101001	Rd	Ra	Imm		$RD \leftarrow RA\ and\ S(Imm)$
XORI	Rd,Ra,Imm	101010	Rd	Ra	Imm		$RD \leftarrow RA\ xor\ S(Imm)$
ANDNI	Rd,Ra,Imm	101011	Rd	Ra	Imm		$RD \leftarrow RA\ and\ \overline{S(Imm)}$
IMM		101100	-	-	Imm		$RIMM \leftarrow Imm$, $MSR[I] \leftarrow 1$

Table 2 – Memory access instructions.

Assembly mnemonic and operands		Instruction word					Operation
		31-26	25-21	20-16	15-11	10–0	
LBU	Rd,Ra,Rb	110000	Rd	Ra	Rb	000 0000 0000	$RD[7:0] \leftarrow M[Ra+Rb]$ $RD[31:8] \leftarrow 0$
LHU	Rd,Ra,Rb	110001	Rd	Ra	Rb	000 0000 0000	$RD[15:0] \leftarrow M[Ra+Rb]$ $RD[31:16] \leftarrow 0$
LW	Rd,Ra,Rb	110010	Rd	Ra	Rb	000 0000 0000	$RD \leftarrow M[Ra+Rb]$
SB	Rd,Ra,Rb	110100	Rd	Ra	Rb	000 0000 0000	$M[Ra+Rb] \leftarrow Rd[7:0]$
SH	Rd,Ra,Rb	110101	Rd	Ra	Rb	000 0000 0000	$M[Ra+Rb] \leftarrow Rd[15:0]$
SW	Rd,Ra,Rb	110110	Rd	Ra	Rb	000 0000 0000	$M[Ra+Rb] \leftarrow Rd[31:0]$
LBUI	Rd,Ra,Imm	111000	Rd	Ra	Imm		$RD[7:0] \leftarrow M[Ra+S(Imm)]$ $RD[31:8] \leftarrow 0$
LHUI	Rd,Ra,Imm	111001	Rd	Ra	Imm		$RD[15:0] \leftarrow M[Ra+S(Imm)]$ $RD[31:16] \leftarrow 0$
LWI	Rd,Ra,Imm	111010	Rd	Ra	Imm		$RD \leftarrow M[Ra+S(Imm)]$
SBI	Rd,Ra,Imm	111100	Rd	Ra	Imm		$M[Ra+S(Imm)] \leftarrow Rd[7:0]$
SHI	Rd,Ra,Imm	111101	Rd	Ra	Imm		$M[Ra+S(Imm)] \leftarrow Rd[15:0]$
SWI	Rd,Ra,Imm	111110	Rd	Ra	Imm		$M[Ra+S(Imm)] \leftarrow Rd[31:0]$

Table 3 – Control instructions

Assembly mnemonic and operands		Instruction word					Operation
		31-26	25-21	20-16	15-11	10–0	
BR	Rb	100110	00000	00000	Rb	000 0000 0000	$PC \leftarrow PC + RB$
BRL	Rd,Rb	100110	Rd	00100	Rb	000 0000 0000	$RD \leftarrow PC\,, PC \leftarrow PC + RB$
BRA	Rb	100110	00000	01000	Rb	000 0000 0000	$PC \leftarrow RB$
BRAL	Rd,Rb	100110	Rd	01100	Rb	000 0000 0000	$RD \leftarrow PC\,, PC \leftarrow RB$
BEQ	Ra,Rb	100111	00000	Ra	Rb	000 0000 0000	$PC \leftarrow PC + RB \; if \; RA = 0$
BNE	Ra,Rb	100111	00001	Ra	Rb	000 0000 0000	$PC \leftarrow PC + RB \; if \; RA \neq 0$
BLT	Ra,Rb	100111	00010	Ra	Rb	000 0000 0000	$PC \leftarrow PC + RB \; if \; RA < 0$
BLE	Ra,Rb	100111	00011	Ra	Rb	000 0000 0000	$PC \leftarrow PC + RB \; if \; RA \leq 0$
BGT	Ra,Rb	100111	00100	Ra	Rb	000 0000 0000	$PC \leftarrow PC + RB \; if \; RA > 0$
BGE	Ra,Rb	100111	00101	Ra	Rb	000 0000 0000	$PC \leftarrow PC + RB \; if \; RA \geq 0$
BRI	Imm	101110	00000	00000	Imm		$PC \leftarrow PC + S(Imm)$
BRLI	Rd,Imm	101110	Rd	00100	Imm		$RD \leftarrow PC\,, PC \leftarrow PC + S(Imm)$
BRAI	Imm	101110	00000	01000	Imm		$PC \leftarrow S(Imm)$
BRALI	Rd,Imm	101110	Rd	01100	Imm		$RD \leftarrow PC\,, PC \leftarrow S(Imm)$
BEQI	Ra,Rb	101111	00000	Ra	Imm		$PC \leftarrow PC + S(Imm) \; if \; RA = 0$
BNEI	Ra,Rb	101111	00001	Ra	Imm		$PC \leftarrow PC + S(Imm) \; if \; RA \neq 0$
BLTI	Ra,Rb	101111	00010	Ra	Imm		$PC \leftarrow PC + S(Imm) \; if \; RA < 0$
BLEI	Ra,Rb	101111	00011	Ra	Imm		$PC \leftarrow PC + S(Imm) \; if \; RA \leq 0$
BGTI	Ra,Rb	101111	00100	Ra	Imm		$PC \leftarrow PC + S(Imm) \; if \; RA > 0$
BGEI	Ra,Rb	101111	00101	Ra	Imm		$PC \leftarrow PC + S(Imm) \; if \; RA \geq 0$

Table 4 – Delayed branch instructions (1 delay slot)

Assembly mnemonic and operands		Instruction word					Operation
		31-26	25-21	20-16	15-11	10–0	
BRD	Rb	100110	00000	10000	Rb	000 0000 0000	$PC \leftarrow PC + RB$
BRLD	Rd,Rb	100110	Rd	10100	Rb	000 0000 0000	$RD \leftarrow PC\,, PC \leftarrow PC + RB$
BRAD	Rb	100110	00000	11000	Rb	000 0000 0000	$PC \leftarrow RB$
BRALD	Rd,Rb	100110	Rd	11100	Rb	000 0000 0000	$RD \leftarrow PC\,, PC \leftarrow RB$
BEQD	Ra,Rb	100111	10000	Ra	Rb	000 0000 0000	$PC \leftarrow PC + RB \text{ if } RA = 0$
BNED	Ra,Rb	100111	10001	Ra	Rb	000 0000 0000	$PC \leftarrow PC + RB \text{ if } RA \neq 0$
BLTD	Ra,Rb	100111	10010	Ra	Rb	000 0000 0000	$PC \leftarrow PC + RB \text{ if } RA < 0$
BLED	Ra,Rb	100111	10011	Ra	Rb	000 0000 0000	$PC \leftarrow PC + RB \text{ if } RA \leq 0$
BGTD	Ra,Rb	100111	10100	Ra	Rb	000 0000 0000	$PC \leftarrow PC + RB \text{ if } RA > 0$
BGED	Ra,Rb	100111	10101	Ra	Rb	000 0000 0000	$PC \leftarrow PC + RB \text{ if } RA \geq 0$
BRID	Imm	101110	00000	10000	Imm		$PC \leftarrow PC + S(Imm)$
BRLID	Rd,Imm	101110	Rd	10100	Imm		$RD \leftarrow PC\,, PC \leftarrow PC + S(Imm)$
BRAID	Imm	101110	00000	11000	Imm		$PC \leftarrow S(Imm)$
BRALID	Rd,Imm	101110	Rd	11100	Imm		$RD \leftarrow PC\,, PC \leftarrow S(Imm)$
BEQID	Ra,Rb	101111	10000	Ra	Imm		$PC \leftarrow PC + S(Imm) \text{ if } RA = 0$
BNEID	Ra,Rb	101111	10001	Ra	Imm		$PC \leftarrow PC + S(Imm) \text{ if } RA \neq 0$
BLTID	Ra,Rb	101111	10010	Ra	Imm		$PC \leftarrow PC + S(Imm) \text{ if } RA < 0$
BLEID	Ra,Rb	101111	10011	Ra	Imm		$PC \leftarrow PC + S(Imm) \text{ if } RA \leq 0$
BGTID	Ra,Rb	101111	10100	Ra	Imm		$PC \leftarrow PC + S(Imm) \text{ if } RA > 0$
BGEID	Ra,Rb	101111	10101	Ra	Imm		$PC \leftarrow PC + S(Imm) \text{ if } RA \geq 0$
RTSD	Ra,Imm	101101	10000	Ra	Imm		$PC \leftarrow RA + S(Imm)$

2. Work assignment

There are two work options, one (Option A) available to all students, the second (Option B) only for student enrolled in the MEIC degree. To guarantee the successful accomplishment of the proposed work, a set of milestones are required for both work assignments. The following paragraphs details the proposed work and presents such milestones.

[Option A] VHDL description of a 5-stage pipeline processor

The first work assignment corresponds to the development of a 5-stage pipeline processor in VHDL supporting the above referred set of instructions, and is mandatory for students enrolled in the Masters in Electrical and Computer Engineering (MEEC) degree. To help successfully accomplish such work, the VHDL files for a 5-stage multi-cycle processor are provided, which fully support all instructions presented in tables 1, 2 and 3. Hence, students are required to make the necessary modifications to the provided VHDL files in order to attain a pipeline execution behaviour.

Work schedule:

- Week 1 (March 1) – Synthesis and implementation of the provided VHDL files in order to unveil the maximum operating frequency of the provided processor.
- Week 2 (March 8) – Analysis of the provided VHDL files and drawing of a detailed schematic of the processor components (to be delivered at the end of the lab). Further planning of the modifications required to attain week 3 milestone.
- Week 3 (March 15) – Modification of the processor in order to fully support a pipeline behaviour, as well as the instructions presented in Table 4. Conflicts may be solved by simply stalling the pipeline.

- Week 4 (March 29) – Support for data forwarding mechanisms and static branch prediction. An exhaustive test will also be performed during the lab in order to guarantee the correct processor behaviour. Such a test will be performed by using a set of previously provided benchmarks.
- April 3 – Delivery of a 6-page (max) report, following the IEEE Transactions formatting instructions (https://www.ieee.org/publications_standards/publications/authors/author_templates.html), with the following contents:
    a) short description of the original processor
    b) Conflict identification procedure
    c) Conflict resolution Procedure
    d) Description of the modifications made to the original processor
    e) Performance comparison between the original 5-stage multi-cycle processor and the 5-stage pipeline processor.
    f) Conclusions

  Additional delivery of all VHDL files and test benches required to validate the processor behaviour.

[Option B] Functional simulator of a MicroBlaze processor

The second optional assignment corresponds to the development of a functional processor simulator (may use a 5-stage pipeline structure, although it is not mandatory), also supporting the full set of described instructions, which work is available only for students enrolled in the Masters in Information Systems and Computer Engineering (MEIC) degree. To help with the development of the proposed work, and in particular with the interpretation of a compiler generated binary file, the sources (in C programming language) of a binary(executable)-to-VHDL converter are provided. Hence, students should make the necessary changes to such file in order to execute the program code. Additionally, the functional simulator should support a “standard output” memory-mapped device, such that all writes to address 0xFFFFFFC0 are sent to the standard output.

Work schedule:

- Week 1 (March 1) – Execution and interpretation of the provided binary-to-VHDL translator.
- Week 2 (March 8) – Planning of the simulator internal structure, including the required data structure. At the end of the class students should deliver a short report (around 1 page) with the diagram of the simulator main routines and data structures.
- Week 3 (March 15) – Design of the main simulator routines and support for all instructions included in table 1 and 2.
- Week 4 (March 29) – Final simulator development. An exhaustive test will also be performed during the lab in order to guarantee the correct simulator behaviour. Such a test will be performed by using a set of previously provided benchmarks.
- April 3 – Delivery of a 6-page (max) report, following the IEEE Transactions formatting instructions (https://www.ieee.org/publications_standards/publications/authors/author_templates.html), with the following contents:
    a) Description of the data structures
    b) Description of the main routines
    c) Analysis of the main difficulties in the development of the functional simulator.

d) Conclusions

Additional delivery of all simulator files and test benches required to validate its correct execution.

The binary-to-VHDL converter is distributed with the auxiliary programs zip file, published in the course webpage under “Aulas de laboratório→Compilador MicroBlaze”. Hence, to access the converter, students should unzip the distributed file and look under folder src/util for the files with name bin2vhd_xxx.c.

3. References

[1] John L. Hennessy, David A. Patterson, “*Computer Architecture – A Quantitative Approach*”, 5th Edition, Morgan Kaufmann, 2011.

[2] Xilinx, “MicroBlaze Processor Reference Guide”, http://www.xilinx.com/support/documentation/sw_manuals/mb_ref_guide.pdf

[3] Peter J. Ashenden, “The VHDL Cookbook”, 1st edition, http://tams-www.informatik.uni-hamburg.de/vhdl/doc/cookbook/VHDL-Cookbook.pdf.

[4] Xilinx, “XST User Guide”, v11.3, 2009, http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_2/xst.pdf