The **Microelectronics Training Center**

The MTC is an initiative within the INVOMEC division

Industrialization & Training in Microelectronics

Lab-exercise

Lab 4:

Design of a microprocessor

Introduction

Overview

Target group: Students

Version: 1.0
Date: 21/03/06
Author: Osman Allam
Modified by: Geert Vanwijnsberghe
History : 1/12/06 : Address width correction

This material was developed with support of the European Social Fund.
ESF: Prevent and combat unemployment by promoting employability, entrepreneurship, adaptability and equal opportunities between women and men, and by investment in people.
http://www.esf-agentschap.be

Introduction

This is an introductory module presenting the specifications and architecture of the microprocessor you are going to build in this lab.

Objectives

After completing this module, the student should be able to:

- Understand the microprocessor features
- Identify and specify the building blocks of the microprocessor

Knowledge background

- Knowledge of von Neumann computer organization

Classification

- Level: 1
- Duration: 30 minutes

The lab

The processor you are going to build is called Micro6.
Micro6 is a simple 32-bit microprocessor based on the von Neumann computer organization. It can perform arithmetic and logic operations on integer data.

Specifications

- Data width = 32 bits
- Address width = 12 bits. That corresponds to an addressing space of 4K
- Simple instruction set (see next section)
- An accumulator to hold ALU results
- Load-store approach for data transfer to and from main memory
- 2 ports register file containing the following registers:
- 28 General purpose registers
- 3 index registers
- 1 stack pointer
- 16 slots deep stack for holding program counter contents during subroutine calls

Instruction set
The instruction set provides 3 groups of instructions according to the functions they serve as follows:

Operate group
This group includes the instructions required to perform arithmetic and logic operations on data that is stored in one of the registers of the register file or the accumulator.

Mnemonic	Function
ADD	addition
SUB	subtraction
INC	increment by 1
DEC	decrement by 1
ZRO	load a registers with 0
AND	logic AND
OR	logic OR
NOT	logic invert
XOR	logic XOR
SRA	Arithmetic shift right by up to 32 bits
SLA	Arithmetic shift left by up to 32 bits
SRL	Logic shift right by up to 32 bits
SLL	Logic shift left by up to 32 bits
RTR	Rotate right by up to 32 bits
RTL	Rotate left by up to 32 bits
CMP	Compare two registers
NLL	Null operation

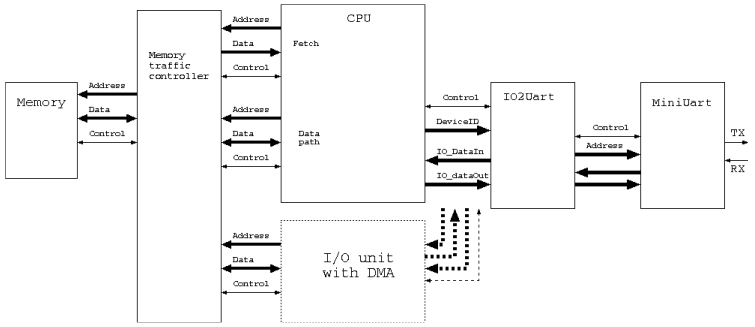
Data transfer group
This group includes the instructions required to move data to and from the main memory. Note that this group implements different addressing modes as will be indicated with each instruction:

Mnemonic	Function	Addressing mode
LD	Load data into a register from the memory location pointed to by another registers	register-indirect
LDX	Load data into a register from the memory location pointed to by the sum of the contents of another register and the contents of an index register	register-indexed
LDM	Load data into a register from a memory location that resides in the first memory page	page-0 instruction
ST	Store the contents of a register into the memory location pointed to by another register	register-indirect
STX	Store the contents of a register into the memory location pointed to by the sum of the contents of another register and the contents of an index register	register-indexed
PSH	Push the contents of a register into the memory stack	stack-register
POP	Pop data from the memory stack into a register	stack-register

Program control group
These are the instructions, controlling the program flow. Conditional branch/jump instructions read the ALU condition flags, which are set by previous operate instructions. CMP instruction can be used for explicit comparison of two registers.

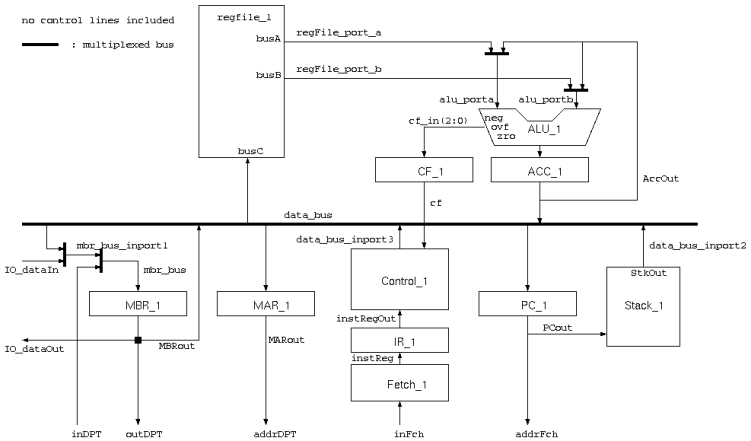
Mnemonic	Function
BRA	Unconditional branch
JSR	Unconditional jump
BGT/JGT	Branch/jump if greater than
BLT/JLT	Branch/jump if less than
BEQ/JEQ	Branch/jump if equal
BNQ/JNQ	Branch/jump if not equal
BGE/JGE	Branch/jump if greater than or equal
BV/JV	Branch/jump if overflow
BNV/JNV	Branch/jump if not overflow
RTN	Return from subroutine
END	End of program

System Architecture block diagram



Note: The DMA controller will not be developed during the lab.

CPU architecture block diagram



Building the Micro6 is done in the following consecutive modules:

- Module 1a: VHDL basics: microprocessor specifications
- Module 1b: VHDL basics: registers
- Module 1c: VHDL basics: counters
- Module 1d: VHDL basics: Multiplexers
- Module 1e: VHDL basics: Register file
- Module 2a: Design of the ALU
- Module 2b: Testbench ALU
- Module 3a: Design of the memory unit
- Module 3b: Design of the memory traffic controller
- Module 3c: Design of the stack
- Module 4a: Design of the decode unit
- Module 4b: Design of the decode-execute unit
- Module 4c: Design of the fetch unit
- Module 5a: Design of the CPU
- Module 5b: Building a basic system
- Module 5c: Transfer assembly code into ram contents
- Module 5d: Simulation of the basic HW-SW system
- Module 6a: Build and simulate a complete system including a Uart
- Module 6b: Implement and verify the complete system including a Uart