# BİLGİSAYAR MİMARİSİ DÖNEM PROJESİ

## Pipelined Processor Implementation

Objectives:

- Using the Logisim simulator
- Designing and testing a Pipelined 16-bit processor

---

## Instruction Set Architecture

In this project, you will design a simple 16-bit RISC processor with seven 16-bit general-purpose registers: R1 through R7. R0 is hardwired to zero and cannot be written, so we are left with seven registers. There is also one special-purpose 16-bit register, which is the program counter (PC). All instructions are 16 bits. There are three instruction formats, R-type, I-type, and J-type as shown below:

**R-type format**

5-bit opcode (Op), 2-bit function field (f), and 3-bit register numbers (Rd, Rs, and Rt)

| $Op^5$ | $f^2$ | $Rd^3$ | $Rs^3$ | $Rt^3$ |
|---|---|---|---|---|

**I-type format**

5-bit opcode (Op), 5-bit immediate constant and 3-bit register number (Rs and Rt)

| $Op^5$ | $Immediate^5$ | $Rs^3$ | $Rt^3$ |
|---|---|---|---|

**J-type format**

5-bit opcode (Op) and 11-bit immediate constant

| $Op^5$ | $Immediate^{11}$ |
|---|---|

For R-type instructions, Rs and Rt specify the two source register numbers, and Rd specifies the destination register number. The function field can specify at most four functions for a given opcode. We can reserve several opcodes for R-type instructions.

For I-type instructions, Rs specifies a source register number, and Rt can be a second source or a destination register number. The immediate constant is only 5 bits because of the fixed-size nature of the instruction. The size of the immediate constant is suitable for our uses. The 5-bit immediate constant can be signed or unsigned depending on the opcode.

For J-type, an 11-bit immediate constant is used for LUI (load upper immediate), J (jump), and JAL (jump-and-link) instructions.

## Instruction Encoding

Nine R-type instructions, twelve I-type instructions, and three J-type instructions are defined. These instructions, their meaning, and their encoding are shown below:

| Instr | Meaning | Encoding | | | | |
|---|---|---|---|---|---|---|
| AND | Reg(Rd) = Reg(Rs) & Reg(Rt) | Op = 00000 | f = 00 | Rd | Rs | Rt |
| OR | Reg(Rd) = Reg(Rs) \| Reg(Rt) | Op = 00000 | f = 01 | Rd | Rs | Rt |
| XOR | Reg(Rd) = Reg(Rs) ^ Reg(Rt) | Op = 00000 | f = 10 | Rd | Rs | Rt |
| NOR | Reg(Rd) = ~(Reg(Rs) \| Reg(Rt)) | Op = 00000 | f = 11 | Rd | Rs | Rt |
| ADD | Reg(Rd) = Reg(Rs) + Reg(Rt) | Op = 00001 | f = 00 | Rd | Rs | Rt |
| SUB | Reg(Rd) = Reg(Rs) – Reg(Rt) | Op = 00001 | f = 01 | Rd | Rs | Rt |
| SLT | Reg(Rd) = Reg(Rs) signed< Reg(Rt) | Op = 00001 | f = 10 | Rd | Rs | Rt |
| SLTU | Reg(Rd) = Reg(Rs) unsigned< Reg(Rt) | Op = 00001 | f = 11 | Rd | Rs | Rt |
| | | | | | | |
| JR | PC = Reg(Rs) | Op = 00010 | f = 00 | 000 | Rs | 000 |
| | | | | | | |
| ANDI | Reg(Rt) = Reg(Rs) & Immediate$^5$ | Op = 00100 | Immediate$^5$ | | Rs | Rt |
| ORI | Reg(Rt) = Reg(Rs) \| Immediate$^5$ | Op = 00101 | Immediate$^5$ | | Rs | Rt |
| XORI | Reg(Rt) = Reg(Rs) ^ Immediate$^5$ | Op = 00110 | Immediate$^5$ | | Rs | Rt |
| ADDI | Reg(Rt) = Reg(Rs) + Immediate$^5$ | Op = 00111 | Immediate$^5$ | | Rs | Rt |
| SLL | Reg(Rt) = Reg(Rs) << Immediate$^4$ | Op = 01000 | Immediate$^5$ | | Rs | Rt |
| SRL | Reg(Rt) = Reg(Rs) zero>> Immediate$^4$ | Op = 01001 | Immediate$^5$ | | Rs | Rt |
| SRA | Reg(Rt) = Reg(Rs) sign>> Immediate$^4$ | Op = 01010 | Immediate$^5$ | | Rs | Rt |
| ROR | Reg(Rt) = Reg(Rs) rot>> Immediate$^4$ | Op = 01011 | Immediate$^5$ | | Rs | Rt |
| LW | Reg(Rt) = Mem(Reg(Rs) + Imm$^5$) | Op = 01100 | Immediate$^5$ | | Rs | Rt |
| SW | Mem(Reg(Rs) + Imm$^5$) = Reg(Rt) | Op = 01101 | Immediate$^5$ | | Rs | Rt |
| BEQ | Branch if (Reg(Rs) == Reg(Rt)) | Op = 01110 | Immediate$^5$ | | Rs | Rt |
| BNE | Branch if (Reg(Rs) != Reg(Rt)) | Op = 01111 | Immediate$^5$ | | Rs | Rt |
| | | | | | | |
| LUI | R1 = Immediate$^{11}$ << 5 | Op = 10000 | Immediate$^{11}$ | | | |
| J | PC = PC + Immediate$^{11}$ | Op = 11110 | Immediate$^{11}$ | | | |
| JAL | R7 = PC + 1, PC = PC + Immediate$^{11}$ | Op = 11111 | Immediate$^{11}$ | | | |

Opcodes 0, 1, and 2 are used for R-type instructions. Opcode 2 is used for the JR (jump register) instruction. Opcodes 4 through 15 are used for I-type instructions.

The 5-bit immediate constant is zero-extended for ANDI, ORI, and XORI. It is sign-extended for the remaining instructions. There are three shift and one rotate instruction. To shift or rotate, use the lower 4 bits of Immediate5 as the shift/rotate amount. There is only one rotate left (ROL) instruction. To rotate right by *n* bits, you can rotate left by 16 – *n* bits, because registers are 16 bits. The Load Upper Immediate (LUI) is of the J-type to have an 11-bit immediate constant loaded into the upper 11 bits of register R1. The LUI can be combined with ORI to load any 16-bit constant into a register. Although the instruction set is reduced, it is still rich enough to write useful programs. We can have procedure calls and returns using the JAL and JR instructions.

## Memory

Your processor will have separate instruction and data memories with 216 words each. Each word is 16 bits or 2 bytes. Memory is *word addressable*. Only words (not bytes) can be read and written to memory, and each address is a word address. This will simplify the processor implementation. The PC contains a word address (not a byte address). Therefore, it is sufficient to increment the PC by 1 (rather than 2) to point to the next instruction in memory. Also, the Load and Store instructions can only load and store words. There is no instruction to load or store a byte in memory.

## Register File

Implement a Register file containing Seven 16-bit registers R1 to R7 with two read ports and one write port. R0 is hardwired to zero.

## Arithmetic and Logical Unit (ALU)

Implement a 16-bit ALU to perform all the required operations:
OR, AND, XOR, NOR, ADD, SUB, SLT, SLTU, SLL, SRL, SRA, ROR

## Addressing Modes

PC-relative addressing mode is used for branch and jump instructions.
For branching (BEQ, BNE), the branch target address is computed as follows:
PC = PC + sign-extend(Imm5), by adding the contents of PC to sign-extended 5-bit Immediate.
For jumps (J and JAL): PC = PC + sign-extend(Imm11).
For LW and SW base-displacement addressing mode is used. The base address in register(Rs) is added to the sign-extended 5-bit immediate to compute the memory address.

## Program Execution

The program will be loaded and will start at address 0 in the instruction memory. The data segment will be loaded and will start also at address 0 in the data memory. To terminate the execution of a program, the last instruction in the program can jump or branch to itself indefinitely.

## Building a Pipelined Processor

It is recommended that you start by building the datapath and control of a single-cycle processor and ensure its correctness. Once you have succeeded in doing this, convert your design and implement a pipelined-datapath and its control logic. A five-stage pipeline should be constructed similar to the pipeline presented in the class lectures. Add pipeline registers between stages. Design the control logic to detect data dependencies among instructions.
For branch and jump instructions, reduce the delay to one cycle only. Stall the pipeline for one clock cycle after a jump or a taken branch instruction. If the branch is not taken, then there is no need to stall the pipeline. Also, stall the pipeline after a LW instruction, if it is followed by a dependent instruction.

## Testing

• Test all components and sub-circuits independently to ensure their correctness. For example, test the correctness of the ALU, the register file, the control logic separately, before putting your components together.
• Test each instruction independently to ensure its correct execution
• Test sequences of dependent instructions. Also, test a LW (load word) followed by a dependent instruction to ensure stalling the pipeline correctly by one clock cycle.
• Test the behavior of taken and untaken branch instructions and their effect on stalling the pipeline.
• Write a sample program that adds an array of integers. Two procedures are required. The main procedure initializes the array elements with some constant values. It then calls the second procedure after passing the array address and the number of elements as parameters in two registers. The second procedure uses the parameters to compute the sum of the array elements and returns the result in a register. Convert the program into machine instructions by hand and load it into the instruction memory starting at address 0. Having two procedures, you will be able to test the JAL and JR instructions.

• Write additional programs as necessary for further testing, translate them by hand, and save them into files. These files can be loaded into the instruction memory and executed. Their data can be saved as well in files and loaded into the data memory.
• Document all your test programs and files and include them in the report document.

---

# Project Report

The report document must contain sections highlighting the following:

## 1 – Design and Implementation

• Specify clearly the design giving detailed description of the datapath, its components, control, and the implementation details (highlighting the design choices you made and why, and any notable features that your processor might have.)
• Provide drawings of the component circuits and the overall datapath.
• Provide a complete description of the control logic and the control signals. Provide a table giving the control signal values for each instruction. Provide the logic equations for each control signal.
• Provide the cases that stall the pipeline, and the logic that you have implemented.
• Carry out the design and implementation with the following aspects in mind:

- Correctness of the individual components
- Correctness of the overall design when wiring the components together
- Completeness: all instructions were implemented properly, stalling the pipeline was handled properly for all cases.

## 2 – Simulation and Testing

• Carry out the simulation of the processor developed using **Logisim.**
• Describe all the features of the simulator used for simulating your design with a clear emphasis on its advantages and limitations (if any) for simulating the design, list the known bugs or missing features (if any).
• Describe the test programs that you used to test your design with enough comments describing the program, its inputs, and its expected output. List all the instructions that were tested and work correctly. List all the instructions that do not run properly.
• Describe all the cases that you handled involving dependences between instructions, cases that stall the pipeline.
• Also provide snapshots of the Simulator window with your test program loaded and showing the simulation output results.
**You will also prepare a live demo for your pipelined processor.**

---

**GRUP ÇALIŞMASI YÖNTEMİ:**

- 5 kişilik gruplar halinde proje hazırlanabilir. Proje raporu ön sayfasına grup üyelerinin isimleri yazılmalıdır.
- Grup üyelerinin her biri projedeki 2 temel aktivite olan
  -Tasarım ve Gerçekleme (Design and Implementation)
  -Benzetim ve Test (- Simulation and Testing)
  faaliyetlerinde rol alacak şekilde koordine olmalıdırlar.
- Proje faaliyetlerini tamamlamak için gereken alt işlemleri zamanlama diyagramı üzerinde bir icra planı hazırlanmalı ve şema üzerinde her bir grup üyesinin yapacağı işler gösterilmelidir.

Ayrıca proje aktivitelerini tartışmak üzere yapılan toplantılar da bu planda yer almalıdır.

---

Proje Teslim Tarihi: **25 Mayıs Çarşamba 2022**