# JURNAL PRAKTIKUM

# OARKOM II

# MODUL II

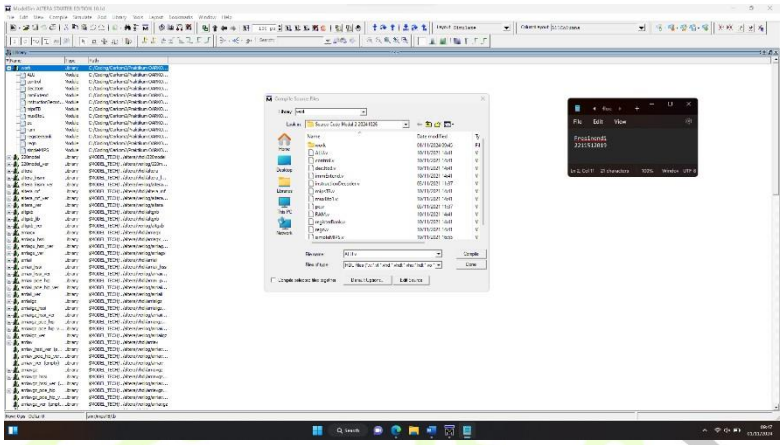
# IMPLEMENTASI PROSESSOR SEDERHANA II


**Nama** : **Fresinendi**
**No. BP** : **2211512019**
**Hari/Tanggal** : **Jumat/ 25 Oktober 2024**
**Shift** :**III**

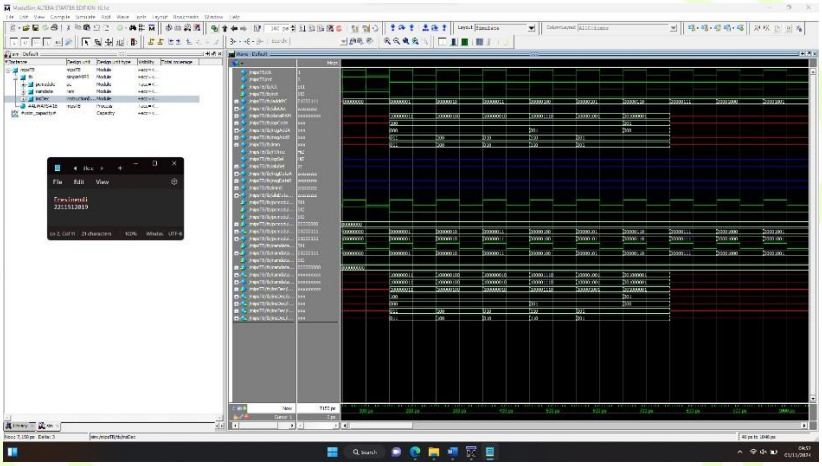
**Asisten** : **1. Mutiara Hikmah**
**2. Lola Dwi Putri**

**LABORATORIUM SISTEM DIGITAL DAN ARSITEKTUR KOMPUTER**
**DEPARTEMEN TEKNIK KOMPUTER**
**FAKULTAS TEKNOLOGI INFORMASI**
**UNIVERSITAS ANDALAS**
**PADANG**
**2024**

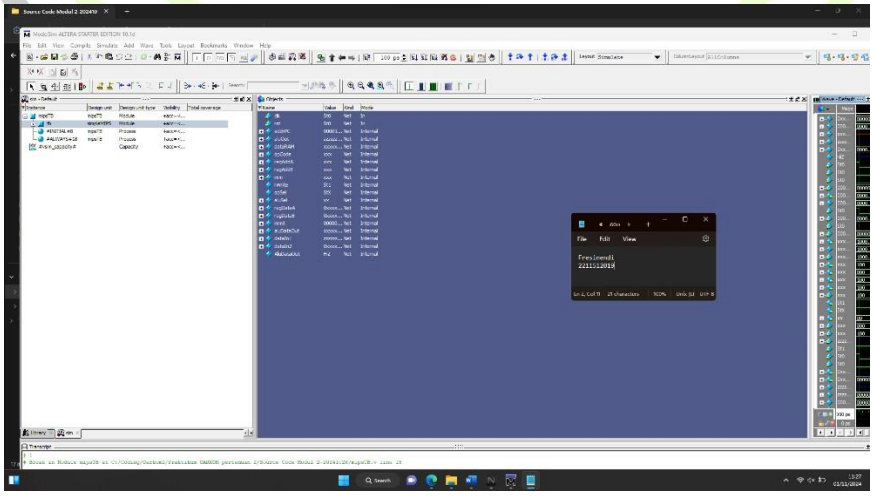
# JURNAL PRAKTIKUM



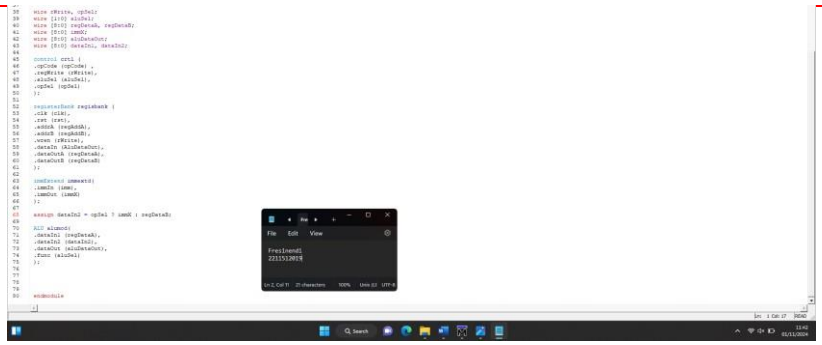
Gambar 1.1 File Sesudah Di Compile



Gambar1.2 Wave Design pada MipsTB.v



Gambar 1.3 Object Pada MipsTB.V



Gambar1.4 Source Kode Program pada Simple MIPS

2.Pertanyaan Praktikum

1.Jelaskan secara runut bagaimana prosesor sederhana ini bekerja ?

2.Bagian apa saja yang diatur oleh control pada prosesor ini ?

3.Apa yang harus dilakukan apabila ingin ditambahkan 4 operasi aritmatika yang baru ?

4.Apa yang harus dilakukan apabila diinginkan prosesor mensupport operasi jump dan branch ?

5.Apa yang harus dilakukan agar lebar data immediate menjadi lebih besar?

**JAWAB**

1. Cara Kerja Prosesor Sederhana

- Fetch: Program counter (pcmodule) menghasilkan alamat instruksi (addrPC) yang digunakan untuk mengambil data instruksi dari memori (RAM) menggunakan ramdata.
- Decode: Decoder instruksi (instructionDecoder) membaca data dari RAM (dataRAM) dan memecahkannya menjadi beberapa bagian, yaitu opCode, regAddA, regAddB, dan nilai langsung (imm).
- Execute: Berdasarkan opCode, modul kontrol akan mengaktifkan sinyal-sinyal yang dibutuhkan (seperti regWrite, aluSel, dan opSel). Register bank (registerBank) mengirimkan data ke ALU sesuai dengan alamat register (regAddA dan regAddB). ALU kemudian menjalankan operasi sesuai instruksi, seperti penjumlahan atau pengurangan.
- Write Back: Hasil dari operasi yang dilakukan oleh ALU disimpan kembali ke register atau memori, tergantung instruksi. Sinyal regWrite memungkinkan data untuk ditulis kembali ke register.

2. Pengaturan Modul Control pada Prosesor

- Modul kontrol berfungsi mengatur sinyal seperti regWrite (untuk mengaktifkan penulisan ke register), aluSel (untuk memilih operasi di

ALU), dan opSel (untuk menentukan apakah data kedua di ALU berasal dari register atau nilai langsung).

- Dalam kode, control menggunakan opCode untuk menentukan sinyal-sinyal tersebut, yang akan memandu alur data antara registerBank, ALU, dan Immediate Extender.

3. Menambahkan Operasi Aritmatika Baru

- Control: Tambahkan pengaturan aluSel baru untuk setiap operasi aritmatika tambahan, misalnya dengan menambah opcode untuk operasi seperti MOD atau MUL.
- ALU: Tambahkan logika dalam ALU untuk mendukung operasi yang baru sesuai dengan instruksi yang diberikan pada aluSel.
- Diagram blok perlu disesuaikan agar ALU dapat menerima sinyal kontrol tambahan dari modul control untuk operasi-operasi yang baru.

4. Menambahkan Dukungan Jump dan Branch

- Diagram dan kode saat ini belum mendukung fungsi jump atau branch, sehingga fitur ini perlu ditambahkan.
- Pada diagram, pcmodule perlu diperbarui agar dapat melakukan override pada addrOut dengan alamat yang ditentukan oleh jump atau branch. Control unit harus mengatur addrIn di pcmodule dalam kondisi tertentu saat diperlukan jump atau branch.
- Di kode, logika tambahan diperlukan untuk mendukung opcode baru (misalnya, BEQ untuk "branch if equal"), dan mungkin juga perlu ditambahkan logika perbandingan di ALU untuk menghasilkan flag.

5. Mempebesar Lebar Data Immediate

- Di kode, immediateExtender saat ini memperluas nilai immediate dari 3-bit menjadi 9-bit. Jika lebar immediate perlu ditingkatkan, modul ini harus disesuaikan.
- Diagram blok menampilkan Immediate Extender sebagai modul terpisah yang dapat diperbesar untuk mendukung immediate yang lebih besar (misalnya, dari 3-bit menjadi 8-bit atau 16-bit). Ini berarti perlu memperbarui ukuran imm di instructionDecoder dan memastikan bahwa ALU dan register bank mampu menangani data dengan lebar yang diperbesar.


DIGIKOM
LAB