MODUL I

Implementasi Prosesor Sederhana I

1.1 TUJUAN

1) Mampu memahami apa itu prosesor sederhana versi arsitektur MIPS dengan format intruksi yang ditetapkan.
2) Mengetahui komponen-komponen *Program Counter*(PC), *Instruction Memory*(IR), dan *Instruction Decoder*(ID).
3) Mampu membuat dan memahami komponen *Program Counter*(PC), *Instruction Memory*(IR), dan *Instruction Decoder*(ID) pada arsitektur MIPS.

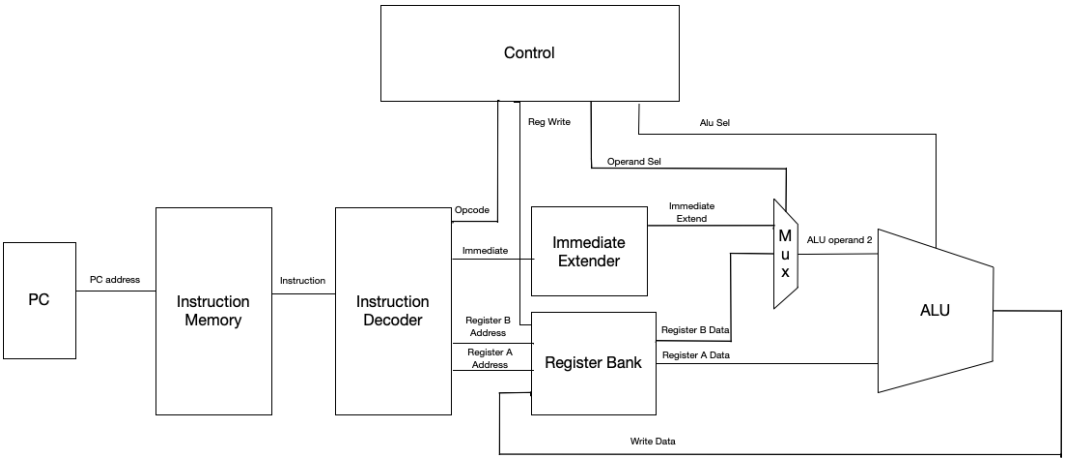
1.2 PERSIAPAN

Pelajari kembali modul praktikum dan bahan kuliah yang berkaitan dengan *Implementasi Prosesor Sederhana I.* Kemudian kerjakan Tugas Pendahuluan dan kumpulkan sesuai ketentuan yang berlaku.

1.3 DASAR TEORI

1.3.1 Prosesor Sederhana

Prosesor yang akan kita bahas dalam praktikum ini adalah prosesor MIPS. Prosesor MIPS, yang dirancang pada tahun 1984 oleh para peneliti di Universitas Stanford, adalah prosesor RISC (Reduced Instruction Set Computer). Dibandingkan dengan rekan-rekan mereka yang berjenis CISC (Complex Instruction Set Computer) seperti prosesor Intel Pentium, prosesor RISC umumnya mendukung instruksi yang lebih sedikit dan jauh lebih sederhana.



Gambar 1.1 Skema Prosesor Sederhana

Prosesor ini adalah versi sederhana dari arsitektur prosesor MIPS. Komponen-komponen yang terdapat pada prosesor ini adalah

• Program counter, menentukan alamat instruksi dari RAM instruksi yang akan dieksekusi oleh prosesor.

• Ram Instruksi (Instruction Memory) menyimpan kode program yang akan dilakukan oleh prosesor dalam format biner.

• Instruksi decoder, Melakukan dekode bacaan data keluaran memory menjadi format-format instruksi prosesor.

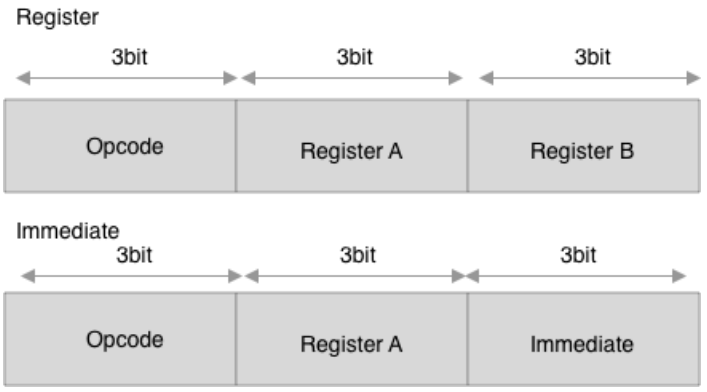
• Control, melakukan kendali datapath pada prosesor sesuai dengan kebutuhan dari instruksi yang akan dijalankan, Pada prosesor sederhana ini control hanya mengendalikan pemilihan operand ALU yang kedua yang dapat berupa immediate atau data dari register B, menentukan jenis operasi alu (Alu Sel), mengatur penulisan register (Reg Write)

• Immediate Extender, melakukan konversi panjang data immediate dari instruksi menjadi sama dengan panjang struktur data Prosesor.

• Register Bank, media penyimpanan register

• ALU, melakukan eksekusi operasi matematika dan logika pada prosesor ini..

Prosesor memiliki Format Instruksi sebagai Berikut



Gambar 1.2 Format Instruksi

Semua hasil operasi akan disimpan pada alamat register A. Opcode hanya 3 bit sehingga hanya mensupport 8 jenis instruksi sebagai berikut

Tabel 1 Opcode Prosesor Sederhana

Opcode	Instruksi
000	Add Tipe Register
100	Add Tipe Immediate
001	Sub Tipe Register
101	Sub Tipe Immediate
010	And Tipe Register
110	And Tipe Immediate
011	Or Tipe Register
111	Or Tipe Immediate

Prosesor memiliki lebar data 9 bit seperti halnya instruksi dan lebar alamat 8 bit, untuk register karena tipe instruksi prosesor hanya mensupport 3 bit maka banyak register adalah 8 buah (R0-R7).

1.3.2 Bahasa Mesin

Bahasa mesin , kode numerik untuk operasi yang dapat dijalankan langsung oleh komputer tertentu . Kode tersebut berupa rangkaian angka 0 dan 1, atau digit biner (“bit”), yang sering diubah dari dan ke heksadesimal (basis 16) untuk dilihat dan

dimodifikasi oleh manusia. Instruksi bahasa mesin biasanya menggunakan beberapa bit untuk merepresentasikan operasi, seperti penjumlahan, dan beberapa untuk merepresentasikan operan, atau mungkin lokasi instruksi berikutnya. Bahasa mesin sulit dibaca dan ditulis, karena tidak menyerupai notasi matematika konvensional atau bahasa manusia, dan kodenya bervariasi dari satu komputer ke komputer lainnya.

Bahasa assembly berada satu tingkat di atas bahasa mesin. Bahasa assembly menggunakan kode mnemonik pendek untuk instruksi dan memungkinkan programmer untuk memasukkan nama untuk blok memori yang menyimpan data.

Contoh: Prosesor akan digunakan untuk melakukan perhitungan berikut

D = (5+2)-(3+1)

Dengan bahasa assembly program di atas dapat dibuat sebagai berikut

addi R0,2. R0 = 2

addi R0,5. R0 = R0+5 (5+2)

addi R1,3. R1 = 3

addi R1,1. R1 = R1 + 1 (1+3)

sub R0,R1. R0 = R0(5+2)-R1(1+3)

Dari bahasa assembly di atas maka dapat dibuat bahasa mesin berdasarkan tabel 1 sebagai berikut:

100000010

100000101

100001011

100001001

001000001

1.4 PERCOBAAN

A. ALAT YANG DIGUNAKAN

1. Quartus II
2. ModelSim - Altera

B. PROSEDUR PERCOBAAN

MEMULAI PERCOBAAN

1. PERCOBAAN 1: Bahasa Assembly dan Bahasa Mesin

Program :

X = (3+4+2) – (6+1)

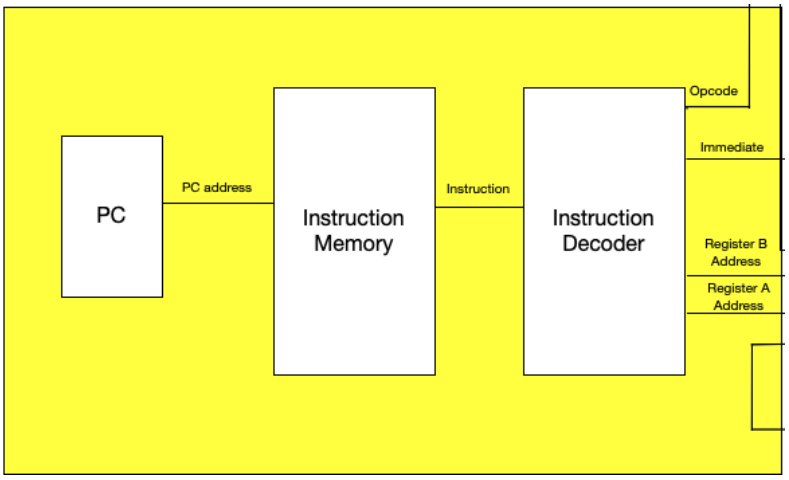
a. Rancanglah bahasa assembly pada jurnal praktikum anda untuk program diatas.

b. Rancanglah bahasa mesin pada jurnal praktikum anda untuk program diatas

c. Lakukan *screenshot* pada hasil rancangan

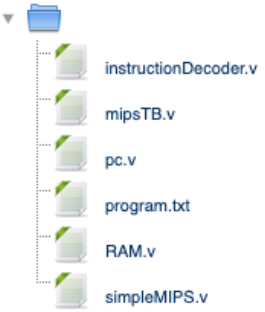
2. PERCOBAAN 2: Implementasi Prosesor dengan Verilog (IF dan ID)

a. Pada modul dirancang bagian PC, memori instruksi dan instruksi dekoder seperti yang terlihat pada gambar di bawah ini



Gambar 1.3 Implementasi Modul 1

b. Download source code yang terdapat pada ilearn



Gambar 1.4 Source Code Modul 1

c. Ganti program pada program.txt dengan bahasa mesin yang anda dapatkan

pada percobaan 1

d. Bukalah program ModelSim Altera

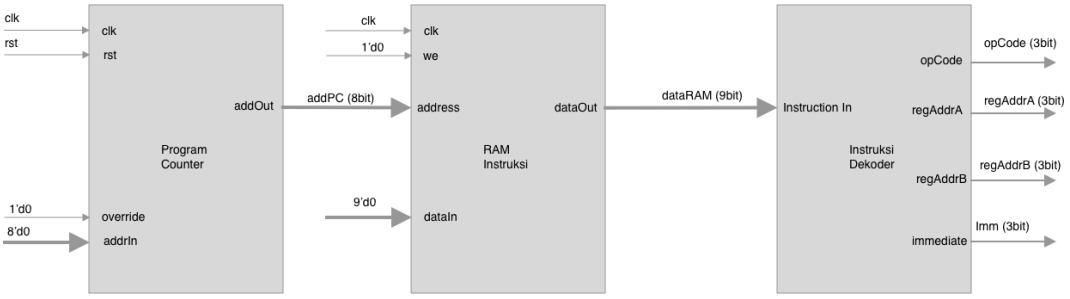
e. Compile file pc.v, RAM.v dan instructionDecoder.v pada modelsim

f. Jelaskan yang dilakukan oleh modul pc.v pada jurnal praktikum.

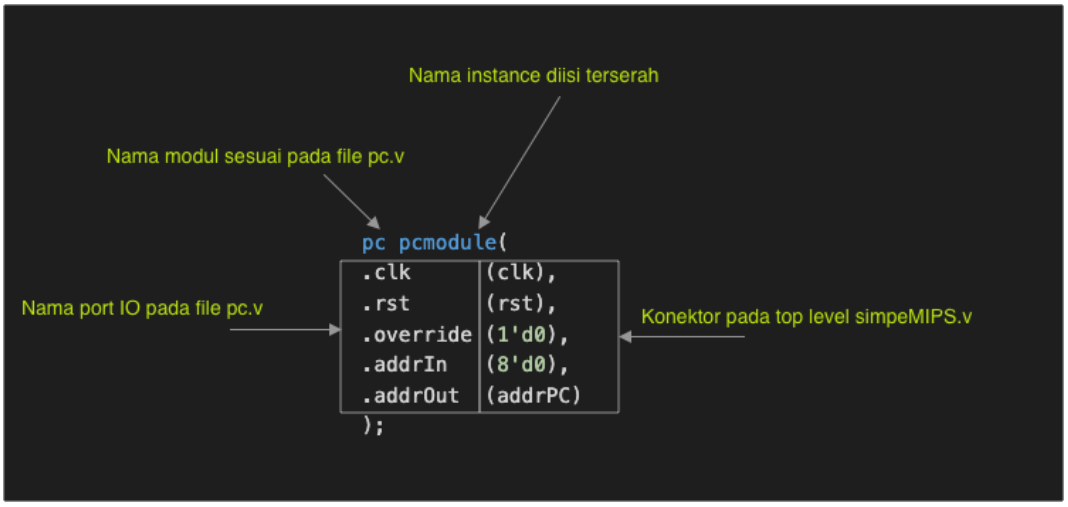
g. Jelaskan yang dilakukan oleh modul RAM.v pada jurnal praktikum.

h. Jelaskan yang dilakukan oleh instructionDecoder pada jurnal praktikum.

i. Buka file simpleMIPS.v dan lanjutkan topLevel MIPS ini sesuai dengan gambar di bawah ini.



Gambar 1.5 Toplevel Instruction Fetch and Instruction Decoder



Gambar 1.6 pemanggilan module pc pada file top level simpleMIPS.v

j. Jalankan test bench mipsTB.v, dengan membuka wave in design dan perhatikan keluaran dari instructiopn decoder apakah telah sesuai dengan yang diharapkan.

k. Lakukan *screenshot* pada hasil praktikum

1.5 PERTANYAAN

1. Jelaskan proses instruction fetch dan instruction decoding secara rinci
2. Jelaskan ada berapa immediate yang dapat disupport oleh prosesor ini, dan bagaimana cara meningkatkannya sehingga prosesor dapat mensupport immediate lebih banyak)
3. Pada prosesor ini apa yang harus dikerjakan oleh immediate extend nantinya.