**Program Studi Teknik Elektro ITB** Nama Kuliah (Kode) : Praktikum Arsitektur Sistem Komputer (EL3111)

Tahun / Semester : 2022-2023 / Ganjil

**Modul** : SYNTHESIZABLE MIPS32® MICROPROCESSOR  
BAGIAN I : INSTRUCTION SET, REGISTER, DAN MEMORY

**Nama Asisten / NIM** :

**Nama Praktikan / NIM** : Ahmad Aziz / 13220034

**Tugas Pendahuluan**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. **Jelaskan bagaimana MIPS32® melakukan eksekusi sebuah instruksi dan jelaskan format tiga instruksi dasar yang dapat dieksekusi oleh MIPS32® beserta penjelasannya untuk setiap bit instruksi! Berikan pula masing-masing lima contoh penggunaan instruksi untuk masing-masing format instruksi dasar!**   Prosesor MIPS32 melakukan eksekusi instruksi melalui 5 tahap sebagai berikut:   1. Instruction Fetch (IF)   Pada tahap pertama ini, prosesor mengatur aliran instruksi yang akan diolah pada tahap berikutnya.   1. Instruction Decode (ID)   Instruksi yang telah di proses pada tahapan sebelumnya atau telah melalui tahapan Fetch akan masuk pada tahapan instruction decode. Pada tahap ini instruksi akan di decode atau dipecah sesuai dengan format instruksi yang digunakan.   1. Execute / Address Calculation (EX)   Pada tahapan ini, dilakukan sebagina besar operasi aritmatika dan logika pada ALU. Selain itu, pada tahap ini juga alamat register diteruskan kembali ke tahap instruction decode sebagai deteksi hazard.   1. Data Memory (MEM)   Pada tahapan ini, terjadi proses pengambilan atau penyimpanan data dari memori.   1. Write Back (WB)   Write back merupakan tahapan terakhir dari proses eksekusi perintah, pada tahapan ini dialirkan data dari memori atau hasil perhitungan dari ALU ke register untuk menjalankan instruksi selanjutnya.   1. **Tentukan nilai opcode dan funct dalam biner, tipe instruksi, dan arti instruksi dari instruksi-instruksi di bawah ini.**   sll sub nor addi xori j  srl and slt slti lui jal  sr or beq andi lw addiu  sdd xor bne ori sw sltiu   |  |  |  |  | | --- | --- | --- | --- | | **instruksi** | **Opcode** | **Syntax** | **Arti** | | sll | 000000 | Shift |  | | srl | 000010 | Shift |  | | sr |  |  |  | | sdd |  |  |  | | sub | 100010 | ArithLog |  | | and | 100100 | ArithLog |  | | or | 100101 | ArithLog |  | | xor | 100110 | ArithLog |  | | nor | 100111 | ArithLog |  | | slt | 101010 | ArithLog |  | | beq | 000100 | Branch |  | | bne | 000101 | Branch |  | | addi | 001000 | ArithLogI |  | | slti | 001010 | ArithLogI |  | | lui |  |  |  | | lw | 100011 | LoadStore |  | | sw | 101011 | LoadStore |  | | j | 000010 | Jump |  | | jal | 000011 | Jump |  | | addiu | 001001 | ArithLogI |  | | sltiu | 001001 | ArithLogI |  |  1. **Diberikan program dalam bahasa assembly berikut ini untuk dieksekusi dalam MIPS32®. Program ini meminta pengguna untuk memasukkan nilai dalam ounce lalu melakukan konversi dari ounce ke pound dan ounce dan menampilkan hasil konversinya. Gunakan teks editor Notepad++ untuk menyalin program ini dan menyimpannya dalam file contoh\_ounces.asm**   # contoh\_ounces.asm  # Konversi dari ounces ke pounds dan ounce.  .data  prompt**:** .asciiz "Masukkan massa dalam ounces: "  pout**:** .asciiz " Pounds\n"  ozout**:** .asciiz " Ounces\n"  **.text**  .globl main  main**:** addu $s0**,** $ra**,** $0 # simpan $31 dalam $16  li $v0**,**4 # tampilkan perintah  la $a0**,**prompt  **syscall**  li $v0**,**5 # baca input pengguna  **syscall**  li $t1**,**16 # 1 pound **=** 16 ounce  divu $v0**,**$t1  mflo $a0  li $v0**,**1 # tampilkan nilai pound  **syscall**  li $v0**,**4 # tampilkan **str** "pounds"  la $a0**,**pout  **syscall**  mfhi $a0 # tampilkan nilai ounce  li $v0**,**1  **syscall**  li $v0**,**4 # tampilkan **str** "ounces"  la $a0**,**ozout  **syscall**  addu $ra**,** $0**,** $s0  jr $ra  # akhir dari program  **a. Simulasikan program tersebut dalam PCSpim lalu screenshot hasil yang ditampilkan dalam console! Perhatikan bahwa PCSpim perlu dikonfigurasi untuk melakukan simulasi menggunakan pseudoinstruction dengan memilih menu Simulator lalu submenu settings. Aktifkan allow pseudo instruction lalu nonaktifkan pilihan bare machine.**    **b. Konversi kode bahasa assembly tersebut ke dalam bahasa C dan lakukan kompilasi menggunakan GCC untuk kemudian dijalankan dalam komputer Anda. Screenshot hasil yang ditampilkan dalam console!.**  #include <stdio.h>  void main**(**void**)** **{**  printf**(**"Masukkan massa dalam ounces: "**);**  int massa **=** 0**;**  scanf**(**"%d"**,** **&**massa**);**  printf**(**"%d Pounds\n"**,** massa**/**16**);**  printf**(**"%d Ounces\n"**,** massa **%** 16**);**  **}**    **c. Bandingkan bahasa assembly program untuk dieksekusi pada mikroprosesor MIPS32® dan bahasa assembly hasil kompilasi oleh GCC untuk dieksekusi pada mikroprosesor Intel® x86. Apa komentar Anda?**   1. **Buatlah program dalam bahasa assembly untuk dieksekusi dalam mikroprosesor MIPS32®. Program ini menerima input berupa total bahan bakar yang dikonsumsi oleh mobil dalam satuan liter dan total jarak yang ditempuh oleh mobil dalam satuan kilometer dengan jumlah bahan bakar tersebut. Kemudian program melakukan perhitungan untuk rata-rata konsumsi bahan bakar per kilometer serta jarak yang dapat ditempuh dalam satuan kilometer 53 menggunakan 1 liter bahan bakar. Sertakan kode bahasa assembly Anda dan screenshot hasil yang ditampilkan pada console.** 2. **Buatlah program dalam bahasa assembly untuk dieksekusi dalam MIPS32® dengan fungsionalitas yang sama dengan program dalam bahasa C berikut ini. Simulasikan program ini dalam PCSpim dan screenshot hasil yang ditampilkan dalam console. (Petunjuk: gunakan bne, beq, atau j untuk merealisasikan loop.**   void main**(){**  int a**;**  int p**;**  int x**;**  a **=** 0**;**  x **=** 1**;**  printf**(**"Masukkan jumlah loop: "**);**  scanf**(**"%d"**,** **&**p**);**  **while** **(**a**<**p**){**  x **=** x **\*** 2**;**  a **=** a **+** 1**;**  **}**  printf**(**"Hasil iterasi: %d\n"**,** x**);**  **}** |