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

Tahun / Semester : 2022-2023 / Ganjil

**Modul** : SYNTHESIZABLE MIPS32® MICROPROCESSOR BAGIAN III : TOP LEVEL DESIGN DAN TESTBENCH

**Nama Asisten / NIM** :

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

**Tugas Pendahuluan**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. **Diberikan sebuah kode program dalam bahasa C sebagai berikut.**   int sum **=** 0**;**  int i **=** 0**;**  **while** **(**i **!=** 10**){**  sum **=** sum **+** 1**;**  i **=** i **+** 1**;**  **}**   1. **Ubahlah kode dalam bahasa C tersebut ke dalam bahasa assembly MIPS32® menggunakan instruksi yang dapat dimengerti oleh mikroprosesor Single-Cycle MIPS32® yang diimplementasikan dalam praktikum ini. Jangan lupa untuk menggunakan instruksi nop setelah instruksi yang berkaitan dengan branching dilaksanakan.**   .data  **.text**  main**:**  li $t1**,** 0 #sum **dec**  li $t2**,** 0 #i **dec**  li $t3**,** 10 #iterate var  **loop:**  beq $t1**,** $t3**,** breakLoop  addi $t2**,** $t2**,** 1  addi $t1**,** $t1**,** 1  j **loop**  breakLoop**:**  **nop**   1. **Ubahlah program bahasa assembly yang telah dibuat pada (a) menjadi file objek yang berisi urutan bilangan biner untuk masing-masing instruksi. Contohnya, apabila terdapat instruksi dalam bahasa assembly addi $s1, $s1, 1, maka kode biner yang bersesuaian adalah 0x22310001.**  |  |  | | --- | --- | | 0x24090000 | li $t1**,** 0 | | 0x240a0000 | li $t2**,** 0 | | 0x240b000a | li $t3**,** 10 | | 0x112b0003 | beq $t1**,** $t3**,** breakLoop | | 0x214a0001 | addi $t2**,** $t2**,** 1 | | 0x21290001 | addi $t1**,** $t1**,** 1 | | 0x08100003 | j **loop** | | 0x00000000 | **nop** |  1. **Diberikan sebuah kode program dalam bahasa C sebagai berikut.**   **.text**  .globl main  .ent main  main**:**  addi $s0**,** $0**,** 19  addi $s1**,** $0**,** 21  bne $s2**,** $s3**,** jumpbne  **nop**  j jumpbne  jumpbne**:**  **sub** $s3**,** $s0**,** $s1  addi $s3**,** $s3**,** 0  addi $s4**,** $s0**,** 4  sw $s1**,** **(**$s4**)**  lw $s5**,** **(**$s4**)**  **add** $s5**,** $s5**,** $0  j end  **nop**  end**:**  li $v0**,** 10  **syscall**   1. **Jelaskan maksud dari bahasa assembly tersebut.**   Pertama, menambahkan 19 dengan memori 0 yang berisi 0 dan dimasukkan pada register $s0  Kemudian juga untuk register $s1 dengan 21  Selanjutnya mengecek apakah $s2 dan $s3 tidak sama, maka masuk ke fungsi  Dalam fungsi, $s0 dikurangi $s1 dan dimasukkan ke $s3  Dan selanjutnya $s4 ditambah 4  Kemudian, memori $s4 diisi dengan $s1 dan $s5 diisi mem $s4  Dan nilai terakhir masuk ke register pc   1. **Ubahlah program bahasa assembly menjadi file objek yang berisi kode biner yang merepresentasikan masing-masing instruksi.**  |  |  | | --- | --- | | **code** | **source** | | 0x20100013 | addi $s0**,** $0**,** 19 | | 0x20110015 | addi $s1**,** $0**,** 21 | | 0x16530002 | bne $s2**,** $s3**,** pc | | 0x00000000 | **nop** | | 0x08100005 | j jumpbne | | 0x02119822 | **sub** $s3**,** $s0**,** $s1 | | 0x22730000 | addi $s3**,** $s3**,** 0 | | 0x22140004 | addi $s4**,** $s0**,** 4 | | 0xae910000 | sw $s1**,** **(**$s4**)** | | 0x8e950000 | lw $s5**,** **(**$s4**)** | | 0x22b50000 | **add** $s5**,** $s5**,** $0 | | 0x0810000d | j | | 0x00000000 | **nop** |  1. **Apa yang dimaksud datapath? Bagaimana hubungan antara datapath dengan control?**   Datapath merupakan komponen dari mikroprosesor yang melakukan operasi aritmetik serta melakukan penyimpanan data. Datapath juga melakukan pemrosesan lima instruksi meliputi instruction fetch, instruction decode, execute, memory access, dan write back dilaksanakan. Sedangkan, control merupakan komponen dari mikroprosesor yang mengatur datapath berdasarkan instruksi yang sedang dieksekusi.   1. **Apabila pada mikroprosesor Single-Cycle MIPS32® yang diimplementasikan dalam praktikum ini diberikan instruksi berikut pada instruction memory**   WIDTH**=**32**;** -- number of bits of data per word  DEPTH**=**65536**;** -- the number of addresses  ADDRESS\_RADIX**=**HEX**;** DATA\_RADIX**=**HEX**;**  CONTENT  **BEGIN**  00 **:** 20100013**;** -- addi 0x10000 0x00000 19  00 **:** ac100000**;** -- sw 0x10000 (0x00000)  **END;**  **Bagaimana datapath eksekusi instruksi tersebut pada mikroprosesor Single-Cycle MIPS32®?**  Isntruksi pada perintah eksekusi tersebut akan menerima code isntruksi yaitu dalam kode diatas adalah isntruksi addi dan sw, berikut juga data yang akan dieksekusi dari dari operan instruksi tersebut. selanjutnya semua data tersebut dimasukkan dalam register yang ditentukan dan control akan menerima instruksi apa yang akan dijalankan beserta nilainya. Eksekusi dilakukan dan output diterima.   1. **Gambarkan dalam bentuk tabel isi instruction memory, data memory, dan register file pada akhir program yang dibuat pada soal 1 dan 2.** 2. **Soal 1**  |  |  |  |  | | --- | --- | --- | --- | | **ins** | **data** | **code** | **source** | | 0x00400000 | 0x00000000 | 0x24090000 | li $t1**,** 0 | | 0x00400004 | 0x00000000 | 0x240a0000 | li $t2**,** 0 | | 0x00400008 | 0x0000000a | 0x240b000a | li $t3**,** 10 | | 0x0040000c | 0x00000003 | 0x112b0003 | beq $t1**,** $t3**,** breakLoop | | 0x00400010 | 0x00000001 | 0x214a0001 | addi $t2**,** $t2**,** 1 | | 0x00400014 | 0x00000001 | 0x21290001 | addi $t1**,** $t1**,** 1 | | 0x00400018 | 0x0040000c | 0x08100003 | j **loop** | | 0x0040001c |  | 0x00000000 | **nop** |  1. **Soal 2**  |  |  |  |  | | --- | --- | --- | --- | | **ins** | **data** | **code** | **source** | | 0x00400000 | 0x00000013 | 0x20100013 | addi $s0**,** $0**,** 19 | | 0x00400004 | 0x00000015 | 0x20110015 | addi $s1**,** $0**,** 21 | | 0x00400008 | 0x00000002 | 0x16530002 | bne $s2**,** $s3**,** pc | | 0x0040000c |  | 0x00000000 | **nop** | | 0x00400010 | 0x00400014 | 0x08100005 | j jumpbne | | 0x00400014 |  | 0x02119822 | **sub** $s3**,** $s0**,** $s1 | | 0x00400018 | 0x00000000 | 0x22730000 | addi $s3**,** $s3**,** 0 | | 0x0040001c | 0x00000004 | 0x22140004 | addi $s4**,** $s0**,** 4 | | 0x00400020 | 0x00000000 | 0xae910000 | sw $s1**,** **(**$s4**)** | | 0x00400024 | 0x00000000 | 0x8e950000 | lw $s5**,** **(**$s4**)** | | 0x00400028 | 0x00000000 | 0x22b50000 | **add** $s5**,** $s5**,** $0 | | 0x0040002c | 0x00400034 | 0x0810000d | j | | 0x00400030 |  | 0x00000000 | **nop** | | 0x00400034 | 0x0000000a | 0x2402000a | li $v0**,** 10 | | 0x00400038 |  | 0x0000000c | **syscall** | |