Akmal Ramadhan - 2206081534 - B - KD

# Tugas 3

Deadline: 28 April 2023, 17.00 Waktu Scele

Pengantar Organisasi Komputer
Semester Genap 2022/2023

**Revisi 1**

---

Petunjuk pengerjaan:

1. Jawaban tugas **HARUS** ditulis tangan atau menggunakan pen tablet. Untuk penulisan tangan dapat ditulis di kertas polos A4 atau kertas folio bergaris dan discan. **Tidak boleh diketik!** Kumpulkan semua halaman jawaban ke satu file yang berformat PDF.
2. Tuliskan **nama, NPM, kelas, dan kode asisten** di **setiap lembar** berkas jawaban Anda. Kelalaian menuliskan keempat informasi ini (lupa atau salah tulis) akan **diberikan penalti -5 poin**/kesalahan.
3. Pastikan tulisan tangan rapi dapat dibaca!
4. Keterlambatan **10 menit < x < 2 jam** dari deadline akan dikenakan penalti **sebesar 30 poin** dari nilai tugas. Keterlambatan **2-6 jam** dari deadline dikenakan penalti **sebesar 60 poin** dari nilai tugas. Pengumpulan yang melewati batasan waktu tersebut **tidak akan dinilai**.
5. Silakan tambahkan asumsi sendiri bila diperlukan.
6. **Plagiarisme adalah pelanggaran serius dengan sanksi nilai 0.**
7. **Warna biru** menunjukan revisi, silahkan tulis pada revisi mana anda mengerjakan, jika tidak menulis maka akan dianggap revisi terbaru.
8. Format penamaan:
   **Tugas3_KodeAsdos_NPM_Nama.pdf**
   Contoh: **Tugas3_RT_1906398364_RicoTadjudin.pdf**

---

#POKoknyaPOK❤️

1. [10] Jelaskan alasan diperlukannya register tambahan (pipeline register) pada datapath untuk setiap tahapan (stage)!

1. Memisahkan stages sehingga setiap tahap dapat beroperasi secara independen
2. Menghubungkan antar dua stages
   Mengeksekusi instruksi secara pipelined dengan menghubungkan output dari satu stage dengan input stage selanjutnya.
3. Melakukan instruksi yang berbeda secara simultan (paralel) atau bersama-sama sehingga dapat meningkatkan performance dan throughput
4. Sinkronisasi clock antara stages memastikan instruksi dieksekusi dengan waktu yang tepat

2. [15] Jelaskan hazard berikut dan berikan contoh solusi untuk mengatasinya!
   a. [5] Control Hazard

Terjadi saat instruksi jump atau branch. Instruksi tidak dapat menentukan apakah harus lompat ke address baru atau tetap di address saat ini. Jika lompat maka dibutuhkan beberapa clock cycles. Penentuan baru diketahui saat stage Memory Access (MEM). Hal ini menyebabkan pipeline menjalani instruksi yang seharusnya tidak dijalankan.
Solusi : Branch prediction yaitu teknik memprediksi instruksi branch.
Jika prediksi benar, instruksi pada jalur yang diprediksi akan dieksekusi.
Jika prediksi salah, instruksi pada jalur sebenarnya akan dieksekusi.

b. [5] Data Hazard

Instruksi membutuhkan data yang sama harus dieksekusi secara berurutan sehingga terjadi konflik data.
Solusi : Forwarding yaitu teknik langsung mengirimkan data yang dihasilkan dari instruksi yang telah dieksekusi ke instruksi berikutnya yang membutuhkan data tanpa harus menunggu data.

c. [5] Structural Hazard

Instruksi berbeda menggunakan komponen yang sama secara bersamaan
contohnya yaitu Instruction Decode (ID) dan Writeback (WB) menggunakan register secara bersamaan.
Solusi : membuat read dan write secara terpisah

3. [30] Peokra ingin membuat suatu arsitektur baru. Namun, karena terbatasnya komponen yang ada, arsitektur tersebut hanya memiliki empat stage, yaitu:
   - Fetch stage
   - Decode stage
   - Execute stage
   - Memory stage

   Agar memudahkan pengguna yang sering memakai arsitektur MIPS, Peokra tetap membagi instruksi yang ada seperti arsitektur MIPS dengan waktu pengolahan instruksi sebagai berikut:

| Instruksi | Fetch Stage | Decode Stage | Execute Stage | Memory Stage | Total Time |
|---|---|---|---|---|---|
| lw | 200 ps | 100 ps | 200 ps | 200 ps | 700 ps |
| sw | 200 ps | 100 ps | - | 300 ps | 600 ps |
| r-format | 200 ps | 100 ps | 200 ps | - | 500 ps |
| beq | 200 ps | 200 ps | - | - | 400 ps |

   Jika Peokra ingin menjalankan kode berikut:

```
lw $t0, 0($s0)
lw $t1, 4($s0)
add $t2, $t0, $t1
lw $t3, 8($s0)
lw $t4, 12($s0)
sub $t5, $t4, $t3
beq $t5, $t2, exit
lw $t6, 16($s0)
mult $t6, $t5
mfhi $t5
mflo $t7
add $t5, $t5, $t2
sub $t7, $t7, $t2
sw $t5, 0($s1)
sw $t7, 4($s1)1
```

Hitunglah berapa lama waktu yang dibutuhkan untuk menjalankan instruksi tersebut menggunakan implementasi (asumsikan beq gagal):

a. [10] Single-cycle

Instruksi yang paling lama : 1 cycle lw → 700 ps

Cycle per instruction : 1

Total instruksi : 15

t = 700 × 1 × 15 = 10500 ps = 10,5 ns

b. [10] Multi-cycle

Tahap yang paling lama : 1 step MEM sw → 300 ps

Cycle per step : 1

t = 300 ps × Σ ( banyak step × banyak instruksi)

= 300 ps × ( 4 × 5 + 3 × 2 + 3 × 7 + 2 × 1 ) = 300 ps × 49

lw sw r-format beq = 14700 ps = 14,7 ns

0$

c. [10] Pipeline (asumsikan tidak ada pipeline hazards)

Tahap yang paling lama : 1 step MEM sw → 300 ps

Cycle per instruction : 4

t = 300 ps × ( CPI + banyak instruksi - 1 )

= 300 ps × (4 + 15 - 1)

= 300 ps × 18 = 5400 ps = 5,4 ns

4. [30] Jika diketahui terdapat prosesor dengan arsitektur MIPS menjalankan kode berikut menggunakan implementasi pipeline (asumsikan tidak ada structural hazard dan hanya terdapat rangkaian forwarding):

```
lw $t1, 0($t0)
lw $t2, 4($t0)
add $t3, $t2, $t1
lw $t4, 8($t0)
sub $t5, $t4, $t3
sw $t5, 0($t1)
lw $t2, 12($t0)
add $t5, $t2, $t4
sub $t6, $t1, $t4
sw $t6, 0($t5)
```

   a. [10] Agar kode di atas dapat berjalan, compiler perlu menambahkan instruksi nop/stall pada kode tersebut. Tuliskan ulang kode tersebut dengan menambahkan instruksi nop/stall secukupnya!

```
lw   $t1, 0 ($t0)
lw   $t2, 4 ($t0)
     nop
add  $t3, $t2, $t1
lw   $t4, 8 ($t0)
     nop
sub  $t5, $t4, $t3
sw   $t5, 0($t1)
lw   $t2, 12 ($t0)
     nop
add  $t5, $t2, $t4
sub  $t6, $t1, $t4
sw   $t6, 0 ($t5)
```

b. [10] Dengan metode Code Scheduling, atur ulang kode tersebut tanpa mengubah logika kode agar jumlah cycle saat menjalankan kode di atas menjadi minimal!

```
lw   $t1, 0 ($t0)
lw   $t2, 4 ($t0)
lw   $t4, 8 ($t0)
add  $t3, $t2, $t1
sub  $t5, $t4, $t3
lw   $t2, 12 ($t0)
sw   $t5, 0($t1)
add  $t5, $t2, $t4
sub  $t6, $t1, $t4
sw   $t6, 0 ($t5)
```

c. [10] Gambarkan implementasi pipeline untuk poin a dan poin b! Gambar tidak diwajibkan dalam bentuk tulis tangan dan dilampirkan pada akhir file pdf

**TERLAMPIR DI AKHIR HALAMAN**

5. [15] Perhatikan dua instruksi pada prosesor dengan arsitektur MIPS berikut.

Instruksi 1: `add $s0, $t0, $t1`
Instruksi 2: `sub $t2, $s0, $t3`

Sebelum dilakukan forwarding, terlebih dahulu akan dilakukan pengecekan terhadap beberapa nilai untuk menentukan apakah perlu dilakukan forwarding atau tidak. Jawablah pertanyaan di bawah berikut berdasarkan penjelasan di atas.

a. [5] Sebutkan control signal yang diperiksa beserta nilainya jika forwarding akan dilakukan!

Control Signal yang harus diperiksa yaitu EX/MEM. Reg Write dan MEM/WB. Reg Write yang nilainya harus 1 karena instruksi sebelumnya menulis ke register dan instruksi berikutnya membutuhkan data dari register yang sama

b. [5] Tentukan register dalam rs, rt, dan rd dari kedua instruksi di atas!

| Instruksi | rs | rt | rd |
|---|---|---|---|
| 1 | $t0 | $t1 | $s0 |
| 2 | $s0 | $t3 | $t2 |

c. [5] Apakah kedua instruksi di atas dapat dilakukan forwarding? Jelaskan! (asumsikan bahwa instruksi 1 dijalankan lebih dahulu dibandingkan instruksi 2)

Ya, kedua instruksi menjalankan forwarding. Instruksi 2 memerlukan nilai dari $s0 di instruksi 1. Karena nilai tersebut sudah dihasilkan di EX stage (dieksekusi ALU), kita bisa langsung pass nilai tersebut ke IF stage instruction 2 tanpa perlu menunggu ke MEM stage.

**Solusi Nomor 4c) - Implementasi Pipeline Nomor 4a)**

| Instruksi | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| lw $t1, 0($t0) | IF | ID | EX | MM | WB | | | | | | | | | | | | |
| lw $t2, 4($t0) | | IF | ID | EX | MM | WB | | | | | | | | | | | |
| nop | | | | | | | | | | | | | | | | | |
| add $t3, $t2, $t1 | | | | IF | ID | EX | MM | WB | | | | | | | | | |
| lw $t4, 8($t0) | | | | | IF | ID | EX | MM | WB | | | | | | | | |
| nop | | | | | | | | | | | | | | | | | |
| sub $t5, $t4, $t3 | | | | | | | IF | ID | EX | MM | WB | | | | | | |
| sw $t5, 0($t1) | | | | | | | | IF | ID | EX | MM | WB | | | | | |
| lw $t2, 12($t0) | | | | | | | | | IF | ID | EX | MM | WB | | | | |
| nop | | | | | | | | | | | | | | | | | |
| add $t5, $t2, $t4 | | | | | | | | | | | IF | ID | EX | MM | WB | | |
| sub $t6, $t1, $t4 | | | | | | | | | | | | IF | ID | EX | MM | WB | |
| sw $t6, 0($t5) | | | | | | | | | | | | | IF | ID | EX | MM | WB |

#POKoknyaPOK❤️

**Solusi Nomor 4c) - Implementasi Pipeline Nomor 4b)**

| Instruksi | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| `lw $t1, 0($t0)` | IF | ID | EX | MM | WB | | | | | | | | | |
| `lw $t2, 4($t0)` | | IF | ID | EX | MM | WB | | | | | | | | |
| `lw $t4, 8($t0)` | | | IF | ID | EX | MM | WB | | | | | | | |
| `add $t3, $t2, $t1` | | | | IF | ID | EX | MM | WB | | | | | | |
| `sub $t5, $t4, $t3` | | | | | IF | ID | EX | MM | WB | | | | | |
| `lw $t2, 12($t0)` | | | | | | IF | ID | EX | MM | WB | | | | |
| `sw $t5, 0($t1)` | | | | | | | IF | ID | EX | MM | WB | | | |
| `add $t5, $t2, $t4` | | | | | | | | IF | ID | EX | MM | WB | | |
| `sub $t6, $t1, $t4` | | | | | | | | | IF | ID | EX | MM | WB | |
| `sw $t6, 0($t5)` | | | | | | | | | | IF | ID | EX | MM | WB |

#POKoknyaPOK❤️