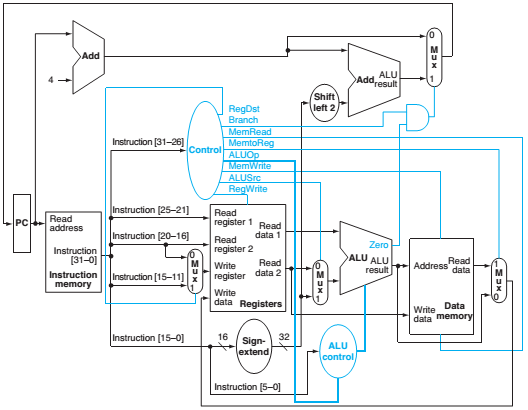
**Bài Tập (Datapath)**

**---oOo---**

**Các bài tập chương này được trích dẫn và biên soạn lại từ:**

***Computer Organization and Design: The Hardware/Software Interface*, Patterson, D. A., and J. L. Hennessy, Morgan Kaufman, Third Edition, 2011.**

**----------------------**



*Hình 1.*

**Bài 1.** **(4.1 – sách tham khảo)**

Cho 2 lệnh như sau:

|  |  |  |
| --- | --- | --- |
|  | **Lệnh** | **Ý nghĩa** |
| a. | add rd, rs, rt | Reg[rd] = Reg[rs] + Reg[rt] |
| b. | lw rt, offs(rs) | Reg[rt] = Mem[Reg[rs]+offs] |

Với từng lệnh trong bảng này:

* 1. Giá trị các tín hiệu điều khiển từ khối “Control” sẽ như thế nào?
  2. Các khối nào trong datapath hình 1 cần thiết, khối nào không cần thiết?
  3. Khối nào trong datapath hình 1 có output đầu ra, nhưng output này không được sử dụng cho lệnh? Khối nào không có output?

Cho thời gian cần để hoàn thành của từng khối trong hình 1 như sau (khối nào không có trong bảng xem như thời gian cần để hoàn thành bằng 0):

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | I-Mem | Add | Mux | ALU | Regs | D-Mem |
| a. | 400ps | 100ps | 30ps | 120ps | 200ps | 350ps |
| b. | 500ps | 150ps | 100ps | 180ps | 220 | 1000ps |

* 1. Tính thời gian cần để hoàn thành lớn nhất của lệnh “and” trong kiến trúc MIPS và cho biết “critical path” của lệnh?

*Chú ý: “Critical path” của một lệnh là đường đi có thời gian trễ lớn nhất trong số các đường có thể khi lệnh thực thi.*

* 1. Tính thời gian cần để hoàn thành lớn nhất của lệnh “lw” trong kiến trúc MIPS và cho biết “critical path” của lệnh?
  2. Tính thời gian cần để hoàn thành lớn nhất của lệnh “beq” trong kiến trúc MIPS và cho biết “critical path” của lệnh?

**---oOo---**

**Bài 2. (4.2 – sách tham khảo)**

Giả sử tập lệnh có thêm hai lệnh mới như sau:

|  |  |  |
| --- | --- | --- |
|  | Lệnh | Ý nghĩa |
| a. | add3 rd, rs, rt, rx | Reg[rd] = Reg[rs] + Reg[rt] + Reg[rx] |
| b. | sll rd, rt, shift | Reg[rd] = Reg[rt] << shift (dịch trái shift bits) |

Với từng lệnh trên:

1. Khối nào đang có trong hình 1 có thể sử dụng cho các lệnh này?
2. Khối mới nào cần được thêm vào?
3. Tín hiệu mới nào cần được thêm vào từ khối “Control” để hỗ trợ?

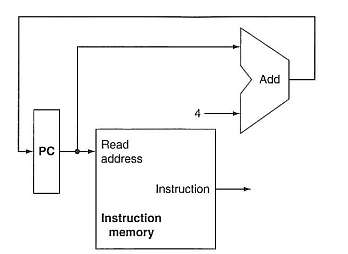
**---oOo---**

**Bài 3. (4.6 – sách tham khảo)**

Giả sử các khối trong datapath (hình 1) có độ trễ như sau:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **I-Mem** | **Add** | **Mux** | **ALU** | **Regs** | **D-Mem** | **Sign-Extend** | **Shift-left-2** |
| a. | 400ps | 100ps | 30ps | 120ps | 200ps | 350ps | 20ps | 2ps |
| b. | 500ps | 150ps | 100ps | 180ps | 220ps | 1000ps | 90ps | 20ps |

1. Giả sử việc duy nhất được thực hiện trong processor chỉ là nạp lệnh liên tục (như hình bên dưới), chu kỳ xung clock cần cho thiết kế là bao nhiêu?



1. Giả sử processor chỉ thực hiện duy nhất mỗi lệnh nhảy (như beq nhưng không cần điều kiện bằng), chu kỳ xung clock cần cho thiết kế là bao nhiêu?
2. Như câu 2, nhưng lệnh nhảy trong trường hợp này có xét đến điều kiện bằng (như beq), chu kỳ xung clock cần cho thiết kế là bao nhiêu?

Cho khối chức năng sau:

|  |  |
| --- | --- |
| a. | Add 4 (bộ cộng dùng để cộng PC với 4) |
| b. | Data Memory |

1. Dạng lệnh nào cần các khối chức năng trên
2. Dạng lệnh nào mà các khối chức năng trên nằm trong critical path?

**---oOo---**

**Bài 4. (4.7 – Sách tham khảo)**

Cho độ trễ của các khối trong datapath như sau:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **I-Mem** | **Add** | **Mux** | **ALU** | **Regs** | **D-Mem** | **Sign-extend** | **Shift-left-2** |
| a. | 400ps | 100ps | 30ps | 120ps | 200ps | 350ps | 20ps | 0ps |
| b. | 500ps | 150ps | 100ps | 180ps | 220ps | 1000ps | 90ps | 20ps |

1. Chu kỳ xung clock là bao nhiêu nếu datapath chỉ hỗ trợ các lệnh thuộc nhóm logic và số học (như add, and, …)?
2. Chu kỳ xung clock là bao nhiêu nếu datapath chỉ hỗ trợ lệnh lw?
3. Chu kỳ xung clock là bao nhiêu nếu datapath hỗ trợ các lệnh: add, beq, lw, sw?

Giả sử tỉ lệ các lệnh được thực hiện trong một đoạn lệnh như sau (Processor không pipeline):

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **add** | **addi** | **not** | **beq** | **lw** | **sw** |
| a. | 30% | 15% | 5% | 20% | 20% | 10% |
| b. | 25% | 5% | 5% | 15% | 35% | 15% |

1. Bao nhiêu phần trăm chu kỳ xung clock có sử dụng khối “Data memory”?
2. Bao nhiêu phần chu kỳ xung clock có sử dụng khối “Sign-extend”?

**---oOo---**

**Bài 5. (4.9 – Sách tham khảo)**

|  |  |
| --- | --- |
|  | Lệnh |
| a. | lw $1, 40($6) |
| b. | label: beq $1, $2, label |

1. Mã máy của hai lệnh trên là gì
2. Chỉ số cung cấp cho input “Read register 1”, “Read register 2” của khối “Registers” là gì? Các thanh ghi này có thật sự được đọc và được sử dụng không? (Xem datapath hình 1)
3. Chỉ số cung cấp cho input “Write register” của khối “Registers” là gì? Thanh ghi này có thật sự được ghi vào không? (Xem datapath hình 1)

**---oOo---**

**Bài 6.**

Cho một kiến trúc máy tính MIPS với datapath và tín hiệu điều khiển như hình.

Đối với lệnh: *addi Rt, Rs, Imm*

1. Lệnh addi chạy được với datapath như trên không? Những khối nào sẽ cần sử dụng cho lệnh trên, khối nào không cần sử dụng?
2. Cho biết giá trị của các tín hiệu điều khiển?
3. Những khối nào có cho dữ liệu output nhưng dữ liệu này không sử dụng? Những khối nào không cho output?
4. Giả sử có lệnh mới như sau “*addi Rt, Rs, Rx, Imm*” *(ý nghĩa Rt = Rs + Rx + Imm)* thì phải thay đổi hay thêm vào hình trên những block nào? (1đ)

**---oOo---**

**Bài 7.**

Một bộ xử lý MIPS 32 bits có datapath như hình và thực thi đoạn chương trình assembly như sau: (Biết khi bắt đầu thanh ghi *$t0 = 0x00000064* và *$t1 = 0x100010FC)*

*or $t9, $zero, $t0*

*add $s0, $zero, $t1*

*sw $t9, 12($s0)*

a. Giá trị output của khối “Instruction memory” là bao nhiêu khi bộ xử lý trên thực thi ở câu lệnh thứ 3?

b. Khi bộ xử lý trên thực thi ở câu lệnh thứ 3, điền các giá trị cho các thanh ghi, tín hiệu điều khiển và các ngõ input/output của datapath theo yêu cầu của bảng sau:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ngõ vào/ra** | | **Điều khiển** | | **Kết quả** | |
| Thanh ghi | Giá trị | Tín hiệu | Giá trị | Ngõ | Giá trị |
| *Instruction[25-21]* |  | *RegDst* |  | *ALUResult*  *(Của ALU)* |  |
| *Instruction [20-16]* |  | *RegWrite* |  | *WriteData*  *(của khối Registers)* |  |
| *Instruction [15-11]* |  | *ALUSrc* |  | *WriteData*  *(Của khối Data Memory)* |  |
| *ReadData1* |  | *Branch* |  |  |  |
| *ReadData2* |  | *MemtoReg* |  |  |  |
|  |  | *MemWrite* |  |  |  |
|  |  | *MemRead* |  |  |  |

**---oOo---**

**Bài 8.**

Một bộ xử lý MIPS 32 bits (có datapath và control như hình) thực thi đoạn chương trình assembly như sau:

*addi $t0, $t1, 8*

*lw $s0, 4($t0)*

*sw $t0, 4($t0)*

Biết khi bắt đầu thanh ghi PC = 0x400000; *$t1 = 0x10010000; $s0 = 0x00000001; word nhớ tại địa chỉ 0x1001000c đang có nội dung (hay giá trị) bằng 0x0000ffff.*

Khi bộ xử lý trên thực thi ở câu lệnh thứ hai, điền các giá trị (tín hiệu, input và output) cho từng khối vào bảng sau:

|  |  |  |
| --- | --- | --- |
| Tên khối | Ngõ | Giá trị |
| **Instruction Memory** | Read address |  |
| Instruction[31-0] |  |
| **Registers** | Read register 1 |  |
| Read register 2 |  |
| Write register |  |
| Write data |  |
| Read data 1 |  |
| Read data 2 |  |
| **ALU** | Input thứ nhất của ALU |  |
| Input thứ hai của ALU |  |
| ALU result |  |
| Zero |  |
| **Data Memory** | Address |  |
| Write data |  |
| Read data |  |
| **Control** | Instruction [31-26] |  |
| RegDst |  |
| Branch |  |
| MemRead |  |
| MemtoReg |  |
| ALUOp **(Chỉ cần cho biết ALU thực hiện phép toán gì)** |  |
| MemWrite |  |
| ALUSrc |  |
| RegWrite |  |

**---oOo---**

**Bài 9.**

Cho một bộ xử lý MIPS 32 bits (có datapath và control như hình).

**Biết *PC = 0x400000*; *$t1 = 0x00008000; $t3 = 0x00000015; Word nhớ tại địa chỉ 0x00008008 có nội dung/giá trị bằng 0x00000015***

Nếu đoạn chương trình sau được thực thi:

***addi $s0, $t1, 4***

***lw $t2, 4($s0)***

***beq $t3, $t2, ABC***

***add $t2, $t3, $t4***

***ABC: sub $t3, $t4, $t5***

Khi bộ xử lý trên thực thi **ở câu lệnh thứ ba**, hỏi:

1. Với khối “Instruction Memory” các ngõ “Read address” và “Instruction[31-0]” bằng bao nhiêu
2. Với khối “Registers”, các ngõ “Read register 1”, “Read register 2”, “Write register”, “Write data”, “Read data 1” và “Read data 2”, “RegWrite” bằng bao nhiêu?
3. Với khối “ALU”, input thứ 1, input thứ hai, “ALU result” và “zero” bằng bao nhiêu?
4. Với khối “Data Memory”, “Address”, “Write data”, “Read data”, “MemWrite”, “MemRead” bằng bao nhiêu?
5. Các tín hiệu điều khiển của 3 MUX: RegDst, ALUSrc và MemToReg bằng bao nhiêu?
6. Đầu vào và đầu ra của khối “Sign-extend” bằng bao nhiêu?
7. Đầu vào và đầu ra của khối “Shift left 2” bằng bao nhiêu?
8. Cổng “AND” trong trường hợp này có kết quả bằng bao nhiêu?
9. Ngõ “ALU Result” của bộ “Add” (mà có một đầu vào là kết quả của “Shift left 2”) có giá trị bao nhiêu?
10. Thanh ghi PC cuối cùng có giá trị bao nhiêu?

**---oOo---**

**Bài 10.**

Cho một processor 16 bit có 4 lệnh như sau

|  |  |  |
| --- | --- | --- |
| Lệnh | Chức năng | Định dạng lệnh |
| add rd, rs, rt | R[rd] = R[rs] + R[rt] | R-format |
| addi rt, rs, imm | R[rt] = R[rs] + SignExt(imm) | I-format |
| lw rt, imm(rs) | R[rt] = M[R[rs] + SignExt(imm)] | I-format |
| bne rs, rt, imm | if(R[rd] != R[rs]) PC = PC + 2 + imm\*2 | I-format |
| **Lưu ý:** rd, rs, rt: thanh ghi imm: số tức thời  R[x]: giá tri thanh ghi x M[y]: nội dung của từ nhớ tại địa chỉ y  SignExt(imm): mở rộng có dấu số tức thời imm từ 4 bit thành 16 bit | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| R-format | opcode | | | | rs | | | | rt | | | | rd | | | |
|  | 15 |  |  | 12 | 11 |  |  | 8 | 7 |  |  | 4 | 3 |  |  | 0 |
| I-format | opcode | | | | rs | | | | rt | | | | imm | | | |
|  | 15 |  |  | 12 | 11 |  |  | 8 | 7 |  |  | 4 | 3 |  |  | 0 |

Với các khối cho sẵn như hình 3, vẽ thêm các đường cần thiết để hoàn chỉnh datapath cho processor có tập lệnh trên (Tại mỗi dấu “?” phải điền vào giá trị tương ứng)

* Có thể dùng thêm bộ MUX, bộ cộng, bộ dịch trái/phải và các loại cổng logic nếu cần
* ALU chỉ nhận input đầu vào là số 16 bit

