**KIẾN TRÚC MÁY TÍNH**
**Khoa Khoa Học và Kỹ Thuật Máy Tính**
**Đại học Bách Khoa – Tp.HCM**
08-2022

Bài tập/Thực hành
**ASSIGNMENT Computer Architecture:**
**Hiện thực máy tính MIPS PIPELINE đơn giản**

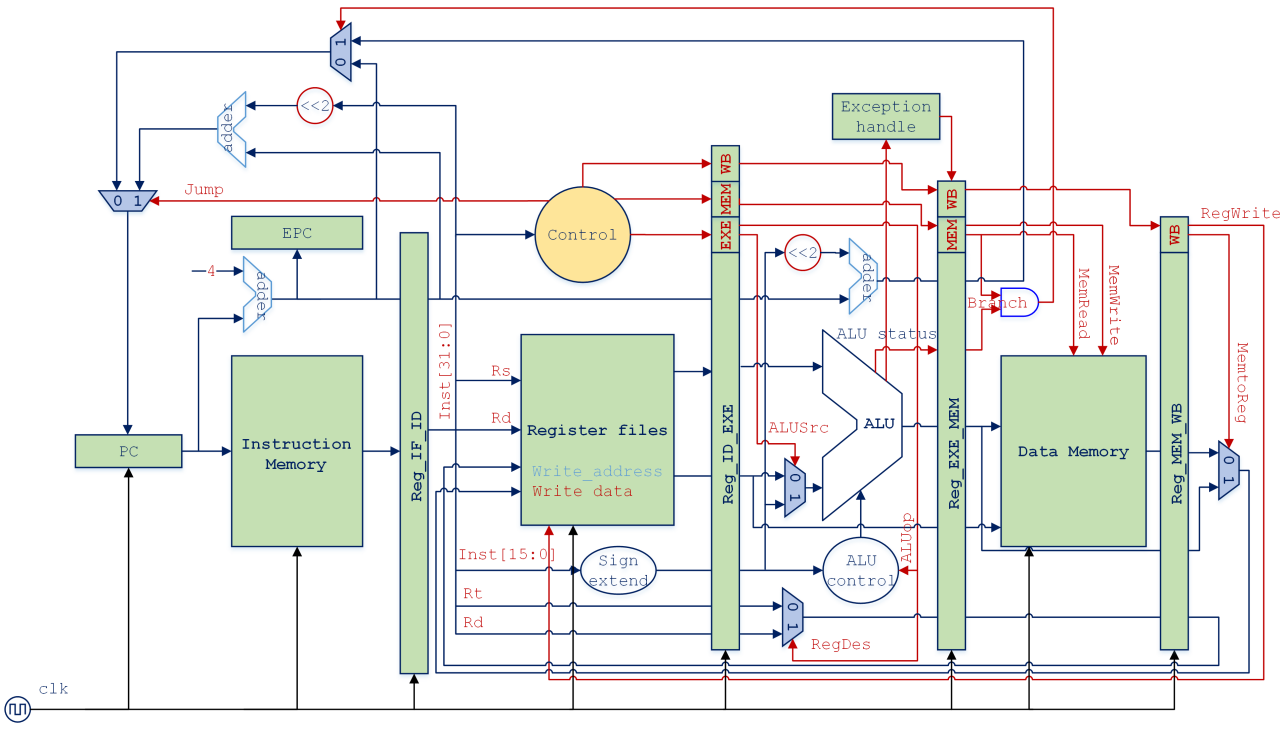
# Yêu cầu chung

- Mỗi nhóm tối đa 4 sinh viên.
- Sử dụng ngôn ngữ Verilog.
- Phần cứng: board DE2i-150 hoặc các board FPGA tương đương.

Bài tập lớn này giúp sinh viên hiểu, phân tích, thiết kế bộ xử lý pipeline. Bên cạnh đó còn giúp sinh viên làm quen với kit FPGA (DE2i-150), làm quen với ngôn ngữ mô tả phần cứng (verilog).

# 1 MIPS Pipeline

Máy tính MIPS Pipeline được mô tả như hình 1.




Hình. 1: Máy tính MIPS Pipeline đơn giản

Kiến trúc trên được chia thành nhiều module và được mô tả ở phần 2

# 2 Đặc tả module

## 2.1 Module system

### 2.1.1 Interface

```
module system (
   input         SYS_clk,
   input         SYS_reset,
   input         SYS_load,
   input [7 :0]  SYS_pc_val,
   input [7 :0]  SYS_output_sel,
   output[26:0]  SYS_leds
);
```

#### 2.1.2 Mô tả

System module là module cao nhất trong phân cấp thiết kế của bộ xử lý. Nó bao gồm các module nhỏ bên trong như: Instruction memory, Register files, ALU, Data memory, Control unit...

- Output của các khối là 32 bits nhưng vì chúng ta chỉ có 27 leds nên tạm thời ta chỉ hiển thị các bit thấp của các output đó.
- Có thể dùng switch để tạo ra xung clk phục vụ trong quá trình demo/debug.
- Khi nhận tín hiệu reset thì hệ thống trở về trạng thái ban đầu (thanh ghi PC reset về zero, các outputs reset về zero ...)
- Ta có thể dùng tín hiệu điều khiển (load) để load 8 bits thấp được nhập từ người dùng vào 8 bits thấp của thanh ghi PC để demo/debug.
- Giá trị output_select được mô tả như sau:
  - 0: leds hiển thị output của khối IM.
  - 1: leds hiển thị output của khối REG.
  - 2: leds hiển thị output của khối ALU.
  - 3: leds hiển thị output của thanh ghi ALU status.
  - 4: leds hiển thị output của khối MEM.
  - 5: leds hiển thị output của khối Control Unit.
  - 6: leds hiển thị output của khối ALU control.
  - 7: leds hiển thị thanh ghi PC, EPC.
  - x: any idea??

### 2.2 Module Instruction memory

#### 2.2.1 Interface

```
module IMEM(
   input [7 :0] IMEM_PC,
   output[31:0] IMEM_instruction
);
```

#### 2.2.2 Mô tả

Bộ nhớ lệnh chứa mã máy chương trình.

### 2.3 Module Register files

#### 2.3.1 Interface

```
module REG(
   input [5:0]  REG_address1,
   input [5:0]  REG_address2,
   input [5:0]  REG_address_wr,
   input        REG_write_1,
   input [31:0] REG_data_wb_in1,
```

```
    output[31:0] REG_data_out1,
    output[31:0] REG_data_out2,
);
```

### 2.3.2 Mô tả

Mạch giải mã thanh ghi, input đầu vào là địa chỉ định danh thanh ghi, output là nội dung của thanh ghi đó. Tín hiệu Regwrite tích cực thì cho phép ghi dữ liệu (data_wr_in) vào thanh ghi tại địa chỉ address_wr.

## 2.4 Bộ ALU

### 2.4.1 Interface

```
module ALU(
    input  [3 :0] ALU_control,
    input  [31:0] ALU_operand_1,
    input  [31:0] ALU_operand_2,
    output [31:0] ALU_result,
    output [7 :0] ALU_status
);
```

### 2.4.2 Mô tả

Là bộ ALU đơn giản chứa các bộ phép tính bên trong (add, sub, and, or, xor, nor, slt , mul, branch, jump...)

Bảng. 1: Tín hiệu thanh ghi trạng thái (register status)

| Tên | Bit | mô tả | Ngoại lệ (exception) |
|---|---|---|---|
| Zero | 7 | Tích cực khi kết quả là zero | |
| overflow | 6 | Tích cực khi bị tràn | x |
| Carry | 5 | Tích cực khi có nhớ | |
| Negative | 4 | Tích cực khi kết quả là số âm | |
| Invalid_address | 3 | Tích cực khi địa chỉ word, haft word không aligned | x |
| Div_zero | 2 | Chia cho zero | x |
| Undefined | [1:0] | Để dành | |

## 2.5 Bộ Data memory

### 2.5.1 Interface

```
module DMEM(
    input [31:0] DMEM_address,
    input [31:0] DMEM_data_in,
    input        DMEM_mem_write,
    input        DMEM_mem_read,
    output[31:0] DMEM_data_out
);
```

### 2.5.2 Mô tả

Là vùng nhớ dữ liệu có kích thước 256 × 32 words. Vì hạn chế về tài nguyên nên ta chỉ lấy 8 bits thấp của phần địa chỉ (địa chỉ 32 bits – là output của ALU) để truy xuất bộ nhớ dữ liệu.

## 2.6 Bộ Control unit

### 2.6.1 Interface

```
module control(
   input [5 :0] opcode,
   output[10:0] control_signal
);
```

#### 2.6.2 Mô tả

Bảng sự thật để sinh ra các tín hiệu điều khiển cho các lệnh dựa vào opcode của các lệnh đầu vào. Tín hiệu control_signal ra được mô tả như bên dưới.

Bảng. 2: Tín hiệu điều khiển

| Tên | bit |
|---|---|
| Jump | 10 |
| Branch | 9 |
| MemRead | 8 |
| MemWrite | 7 |
| Mem2Reg | 6 |
| ALUop | [4:5] |
| Exception | 3 |
| ALUsrc | 2 |
| RegWrite | 1 |
| RegDst | 0 |

Jump 10
Branch 9
MemRead 8
MemWrite 7
ALUOp [6:5]
ALUSrc 4
RegDst 3
Exception 2
Mem2Reg 1
RegWrite 0

### 2.7 Các module khác

#### 2.7.1 Bộ ALU control

Bảng sự thật, input đầu vào là 2 bits Opcode và 6 bits function. Khi đó bộ ALU control sẽ tra bảng và đưa ra tín hiệu chọn để chọn phép tính tương ứng của lệnh hiện tại.(tham khảo textbook để xác định tín hiệu output ALUcontrol).
Trong bài tập lớn này format của các lệnh shift dùng thanh ghi Rs, Rt thay vì Rt, Rd như trong textbook. Mỗi nhóm đưa ra sự định nghĩa của nhóm mình về ALUcontrol cho nhóm lệnh shift, phép nhân, phép chia.

#### 2.7.2 Module exception handle

Khi phát hiện ra các ngoại lệ (exceptions) module này sẽ sinh ra tín hiệu để disable các tín hiệu MemWrite, RegRead nhằm bảo vệ kết quả của hệ thống. Khi xảy ra ngoại lệ thì hệ thống sẽ tạm ngưng hoạt động cho đến khi reset lại trạng thái ban đầu. Trong bài tập lớn này chúng ta xử lý 3 loại ngoại lệ:

- Không xác định được lệnh (undefined instruction).
- Lỗi truy xuất phần cứng (hardware error – ghi vô thanh ghi $zero, canh lề (align) địa chỉ word sai).
- Arithmetic exception (các ngoại lệ về số học như tràn, chia cho zero...).

#### 2.7.3 Thanh ghi Reg_IF_ID

Lưu kết quả trung gian trong quá trình Pipeline.

#### 2.7.4 Thanh ghi Reg_ID_EXE

Lưu kết quả trung gian trong quá trình Pipeline.

#### 2.7.5 Thanh ghi Reg_EXE_MEM

Lưu kết quả trung gian trong quá trình Pipeline.

#### 2.7.6 Thanh ghi Reg_MEM_WB

Lưu kết quả trung gian trong quá trình Pipeline.

#### 2.7.7 Module adder

Bộ cộng.

#### 2.7.8 Module mux

Bộ lựa chọn, tín hiệu chọn được nêu rõ trong hình **??**.

#### 2.7.9 Sign extend

Bộ mở rộng dấu, input là số nguyên 16 bits, 5 bits shamt, output là số nguyên đó nhưng được mở rộng dấu thành 32 bits.

#### 2.7.10 Shift left 2

Bộ dịch 2 bits, thực chất nối dây sao cho tín hiệu lệch đi 2 bit.

#### 2.7.11 Thanh ghi EPC

Thanh ghi đánh dấu lại lệnh tiếp theo khi máy tính thực thi lệnh và báo ngoại lệ (exceptions).

## 3 Yêu cầu hiện thực

- Hiện thực hệ thống trên với 1 số yêu cầu sau:
  - Khi reset hệ thống thì thanh ghi PC trở về zero, output leds được reset về zero, có thể dùng toggle switch để làm xung clock khi demo.
  - Sinh viên tự viết đoạn code bằng hợp ngữ, sau đó dùng công cụ để chuyển sang mã máy (biểu diễn ở dạng hex, có thể dùng MARS 4.4, nhưng chú ý thứ tự các trường trong mã lệnh). Nạp mã máy đó vào Instruction memory để demo.
  - Khi demo cần chỉ rõ output của từng khối( IM, REG, ALU, Control unit, DATA memory) thông qua bộ leds. Dùng các switch để chọn output của các khối cần hiển thị.
  - Thanh ghi PC có thể được load theo ý người dùng thông qua switch và cần chỉ rõ giá trị thanh ghi PC qua leds.
- Xác định cycle của hệ thống mình thiết kế.
- Xác định tài nguyên tổng hợp của hệ thống. Nêu rõ dùng kit nào, công cụ gì...
- Mọi thắc mắc được thảo luận trên forum.
- Nhóm có thể thay đổi đặt tả hay interface của hệ thống/module. Khi thay đổi cần nói rõ lý do, trình bày sự thay đổi đó

Demo mỗi nhóm tự chuẩn bị. Yêu cầu phải thể hiện được các phép toán số học (cộng, trừ, nhân, chia, shift, phép logic), truy xuất vùng nhớ, nhóm lệnh điều khiển (branch, jump), xử lý lỗi.

### 3.1 Điểm thưởng (+1.0đ)

Dùng LCD để hiển thị thông tin sau:

- PC: giá trị hiện tại của thanh ghi PC.
- Out_sel: chọn giá trị khối cần output(được mô tả trong top module).
- Output_val: giá trị output của khối được chọn.

Cả nhóm sẽ nhận điểm 0 (Zero) nếu phát hiện gian lận