Instructions for Lab#5

Student ID: 23133021 Group:

Full in name: Nguyễn Ngọc Hải

Theory (Memory, Architectures, Interrupts and Stacks)

1. Review the lecture slides on types of memory and provide a short answers to the following questions (using your own words):

1.1. What is ROM and what is its primary purpose ?

ROM là viết tắt của Read Only Memory, dịch ra tiếng Việt là bộ nhớ chỉ đọc. Đây là một loại bộ nhớ máy tính mà dữ liệu đã được ghi vào từ trước (thường là bởi nhà sản xuất) và không thể thay đổi hoặc xóa bởi người dùng thông thường.

Mục đích chính của ROM là lưu trữ các chương trình và dữ liệu quan trọng, cần thiết cho việc khởi động và vận hành thiết bị. Có thể hiểu ROM như một "khuôn mẫu" cố định, chứa những hướng dẫn cơ bản để thiết bị "biết" cách hoạt động ngay từ khi được bật lên.

1.2. What is RAM and how is it different from ROM ?

RAM: Bộ nhớ tạm thời, dùng để lưu trữ dữ liệu và chương trình đang hoạt động. Giống như "bàn làm việc" của máy tính, nơi xử lý thông tin hiện hành.

Đặc điểm	RAM	ROM
Tính chất	Khả biến (mất dữ liệu khi tắt nguồn)	Không khả biến (giữ lại dữ liệu ngay cả khi tắt nguồn)
Đọc/Ghi	Đọc và ghi	Chủ yếu chỉ đọc
Chức năng	Lưu trữ dữ liệu và chương trình đang sử dụng	Lưu trữ các hướng dẫn khởi động và chương trình cơ sở cần thiết
Tốc độ	Nói chung nhanh hơn ROM	Nói chung chậm hơn RAM
Chi phí	Thường đắt hơn ROM	Thường rẻ hơn RAM

1.3. What is the difference between static RAM and dynamics RAM ?

SRAM (Static RAM) và DRAM (Dynamic RAM) là hai loại RAM khác nhau, với các đặc điểm sau:

Cấu trúc bên trong:

- SRAM: Sử dụng các mạch latch (khóa) để lưu trữ bit dữ liệu.
- DRAM: Sử dụng tụ điện để lưu trữ bit dữ liệu.

Tốc độ truy xuất:

- SRAM: Tốc độ truy xuất nhanh hơn DRAM.
- DRAM: Tốc độ truy xuất chậm hơn SRAM.

Yêu cầu về nguồn điện:

- SRAM: Không cần cấp điện liên tục để duy trì dữ liệu.
- DRAM: Cần được cấp điện liên tục để duy trì dữ liệu.

Mật độ lưu trữ:

- SRAM: Mật độ lưu trữ thấp hơn DRAM.
- DRAM: Mật độ lưu trữ cao hơn SRAM.

Ứng dụng:

- SRAM: Thường được sử dụng trong bộ nhớ cache của CPU.
- DRAM: Thường được sử dụng làm bộ nhớ chính (main memory) trong máy tính.

1.4. What type of memory is typically used in USB thumb drives ? Why shouldn't we rely on this for critical data storage ?

- Loại bộ nhớ thường được sử dụng trong ổ đĩa USB là Flash Memory, cụ thể là NAND Flash.
- NAND Flash không cần nguồn điện để duy trì dữ liệu (non-volatile), có kích thước nhỏ gọn, dễ di chuyển tốc độ truy xuất nhanh hơn ổ cứng truyền thống.
- Tuy nhiên, NAND Flash cũng có một số hạn chế:

+ Tuổi thọ giới hạn: Mỗi ô nhớ NAND Flash chỉ có thể được ghi/xóa một số lần giới hạn (thường khoảng 10,000 - 100,000 lần). Sau đó, ô nhớ sẽ bị hỏng.

+ Dữ liệu có thể bị mất: Dữ liệu lưu trữ trên NAND Flash có thể bị mất hoặc hỏng do các yếu tố như nhiệt độ, độ ẩm, va đập, từ trường, v.v.

- Vì những lý do trên, không nên dựa vào ổ đĩa USB (NAND Flash) để lưu trữ dữ liệu quan trọng, nhạy cảm hoặc cần được bảo vệ lâu dài. Thay vào đó, nên sử dụng các loại bộ nhớ khác như ổ cứng (HDD) hoặc ổ SSD, có tuổi thọ và độ tin cậy cao hơn.

2. Consider a computer with 1GB RAM (1024 MB). Given memory addressing is for each byte, how many bits are needed to address all bytes in the system's RAM ?

1GB = 1024 MB

1 MB = 1024 KB

1 KB = 1024 byte

Vậy 1GB = 1024 x 1024 x 1024 byte = 2x1030 byte

Để địa chỉ hóa tất cả các byte trong 1GB RAM, ta cần sử dụng số bit đủ lớn để biểu diễn tối đa 1,073,741,824 byte.

Số bit cần thiết được tính như sau:

Số bit = log2(1,073,741,824) = 30 bit

Vì vậy, để địa chỉ hóa tất cả các byte trong 1GB RAM, cần sử dụng 30 bit

3. Give a brief description of the Von Neumann and Harvard computing architectures. What are the fundamental differences between the two and for what is is each designed to achieve ?

- Kiến trúc Von Neumann: Đây là kiến trúc máy tính phổ biến nhất, được thiết kế bởi John von Neumann. Trong kiến trúc này, CPU, bộ nhớ chính và thiết bị nhập/xuất được kết nối với nhau thông qua một bus dữ liệu chung. CPU sử dụng cùng một bộ nhớ để lưu trữ cả dữ liệu và chương trình.

+ Ưu điểm: Đơn giản, dễ thiết kế và lập trình.

+ Nhược điểm: Tốc độ truy xuất bộ nhớ có thể bị giới hạn do sử dụng chung bus dữ liệu.

- Kiến trúc Harvard: Được thiết kế để cải thiện hiệu suất so với kiến trúc Von Neumann. Trong kiến trúc này, CPU có hai bộ nhớ riêng biệt: một bộ nhớ cho dữ liệu và một bộ nhớ cho chương trình. Hai bộ nhớ này được kết nối với CPU thông qua các bus riêng biệt, cho phép truy xuất dữ liệu và chương trình đồng thời.

+ Ưu điểm: Tốc độ truy xuất bộ nhớ nhanh hơn do sử dụng các bus riêng biệt.

+ Nhược điểm: Phức tạp hơn trong thiết kế và lập trình.

- Sự khác biệt cơ bản giữa hai kiến trúc này là:

+ Kiến trúc Von Neumann sử dụng chung bộ nhớ cho dữ liệu và chương trình, trong khi kiến trúc Harvard sử dụng hai bộ nhớ riêng biệt.

+ Kiến trúc Harvard được thiết kế để tăng tốc độ truy xuất bộ nhớ, trong khi kiến trúc Von Neumann có cấu trúc đơn giản hơn.

What is cache memory and what is its primary role ?.

- Bộ nhớ cache là một loại bộ nhớ nhanh, có dung lượng nhỏ, được đặt gần với CPU nhằm giảm

thời gian truy xuất dữ liệu từ bộ nhớ chính (RAM).

- Vai trò chính của bộ nhớ cache là:

+ Tăng tốc độ truy xuất dữ liệu: Khi CPU cần truy xuất dữ liệu, nó sẽ kiểm tra xem dữ liệu đó có trong bộ nhớ cache không. Nếu dữ liệu có trong cache (cache hit), CPU sẽ lấy dữ liệu từ cache, giúp tăng tốc độ truy xuất. Nếu dữ liệu không có trong cache (cache miss), CPU sẽ lấy dữ liệu từ bộ nhớ chính (RAM), tốc độ truy xuất sẽ chậm hơn.

+ Giảm tải cho bộ nhớ chính: Bộ nhớ cache lưu trữ các dữ liệu được truy xuất gần đây, giúp giảm số lần truy xuất vào bộ nhớ chính. Điều này giúp giảm tải cho bộ nhớ chính, tăng hiệu suất của hệ thống.

4. Explain the concept of an interrupt, and list four common types.

- Trong hệ thống máy tính, ngắt (interrupt) là một tín hiệu báo hiệu cho CPU rằng có một sự kiện cần được chú ý ngay lập tức. Sự kiện này có thể đến từ phần cứng (như nhấn phím, di chuyển chuột) hoặc phần mềm (như lỗi chương trình, yêu cầu dịch vụ hệ thống). Khi nhận được ngắt, CPU sẽ tạm dừng công việc hiện tại, lưu trạng thái hiện tại và chuyển sang xử lý sự kiện gây ra ngắt. Sau khi xử lý xong, CPU sẽ khôi phục lại trạng thái trước đó và tiếp tục công việc.

- Bốn loại ngắt phổ biến:

+ Ngắt phần cứng (Hardware Interrupt): Do các thiết bị phần cứng bên ngoài tạo ra.

+ Ngắt phần mềm (Software Interrupt): Do các chương trình phần mềm tạo ra.

+ Ngắt ngoại lệ (Exception): Là một loại ngắt phần mềm đặc biệt, xảy ra khi có lỗi trong quá trình thực thi chương trình.

+ Ngắt định thời (Timer Interrupt): Do bộ đếm thời gian bên trong hệ thống tạo ra, được sử dụng để lên lịch các tác vụ.

4.1. Polling is an alternative to interrupts ? Briefly explain polling and why it is not com- monly used.

Polling là phương pháp mà CPU sẽ liên tục kiểm tra trạng thái của các thiết bị ngoại vi theo một chu kỳ định sẵn. Nếu một thiết bị cần được chú ý (ví dụ như có dữ liệu mới từ bàn phím), CPU sẽ dừng việc đang làm và xử lý yêu cầu của thiết bị đó.

Tại sao polling không phổ biến?

+ Lãng phí tài nguyên CPU: CPU phải dành nhiều thời gian để kiểm tra các thiết bị, kể cả khi chúng không có yêu cầu gì. Điều này làm giảm hiệu năng của hệ thống.

+ Thời gian phản hồi chậm: CPU chỉ kiểm tra thiết bị theo chu kỳ, nên có thể bỏ lỡ các sự kiện xảy ra giữa các lần kiểm tra.

+ Khó mở rộng: Khi số lượng thiết bị tăng lên, thời gian dành cho polling cũng tăng theo, khiến hệ thống trở nên kém hiệu quả.

5. Explain the general concept of a stack - how do they work, and what is their primary pur- pose.

- Stack là một cấu trúc dữ liệu tuyến tính, hoạt động theo nguyên tắc LIFO (Last-In, First-Out), nghĩa là phần tử được thêm vào cuối cùng sẽ là phần tử được lấy ra đầu tiên.

- Cách thức hoạt động:

Stack có hai thao tác cơ bản:

Push (Đẩy): Thêm một phần tử vào đỉnh stack.

Pop (Lấy ra): Lấy ra phần tử ở đỉnh stack.

- Mục đích chính: Stack được sử dụng rộng rãi trong lập trình và hệ thống máy tính với nhiều mục đích, bao gồm:

+ Quản lý bộ nhớ: Stack được sử dụng để quản lý vùng nhớ cho các biến cục bộ, tham số hàm và địa chỉ trả về khi chương trình được thực thi.

+ Xử lý ngắt: Khi một ngắt xảy ra, CPU sẽ lưu trạng thái hiện tại vào stack trước khi chuyển sang xử lý ngắt. Sau khi xử lý xong, CPU sẽ khôi phục trạng thái từ stack.

+ Đánh giá biểu thức: Stack được sử dụng để đánh giá các biểu thức toán học, đặc biệt là biểu thức hậu tố (Reverse Polish Notation).

+ Backtracking (Quay lui): Trong các thuật toán tìm kiếm và duyệt, stack được sử dụng để lưu trữ các trạng thái đã duyệt qua, cho phép quay lui về trạng thái trước đó nếu cần.

+ Triển khai các hàm đệ quy: Stack được sử dụng để lưu trữ các thông tin về mỗi lần gọi hàm đệ quy, cho phép hàm thực hiện đúng và quay trở lại vị trí gọi ban đầu.

5.1. How are stacks useful for handling interrupts ?
Stacks are useful for handling interrupts by maintaining the context of the CPU before an interrupt occurs. When an interrupt is triggered, the current state (including the program counter and registers) is pushed onto the stack

Stack đóng vai trò rất quan trọng trong việc xử lý ngắt, đảm bảo hệ thống có thể phản ứng với các sự kiện bất ngờ mà không làm mất dữ liệu hay gây ra lỗi. Khi một ngắt xảy ra, CPU cần tạm dừng chương trình hiện tại để xử lý ngắt. Trước khi chuyển sang xử lý ngắt, CPU sẽ lưu trữ các thông tin quan trọng về trạng thái của chương trình hiện tại vào stack. Sau khi xử lý xong ngắt, CPU sẽ lấy lại các thông tin đã lưu trên stack để khôi phục trạng thái của chương trình trước đó. Nhờ đó, chương trình có thể tiếp tục thực thi từ vị trí bị gián đoạn mà không gặp vấn đề gì.

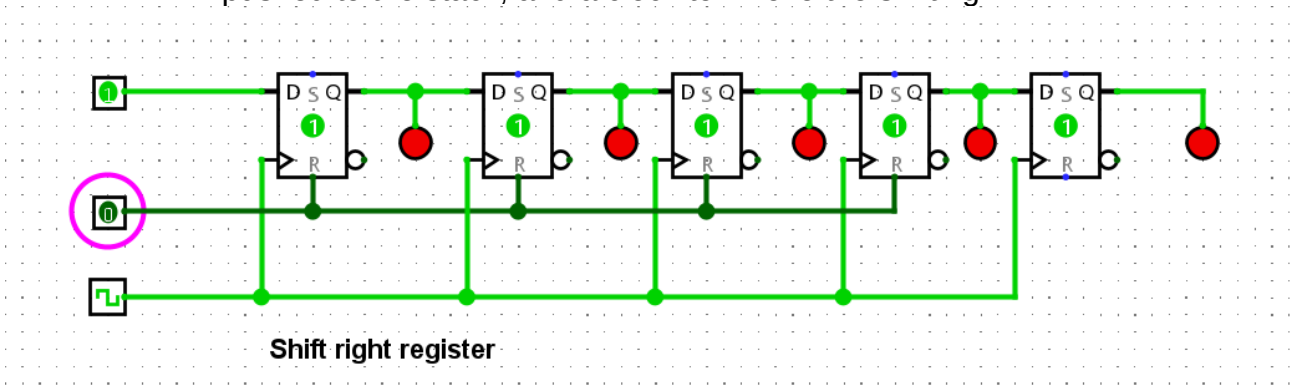
5.2. How are stacks useful in programming ?
Stacks are essential in programming for managing function calls, supporting recursion, implementing undo features, evaluating expressions, and facilitating backtracking by efficiently tracking temporary data and control flow

Ở trong lập trình, Stack hỗ trợ lưu trữ biến cục bộ, tham số hàm, và địa chỉ trả về, đảm bảo chương trình hoạt động trơn tru. Nó cũng hữu ích trong việc tính toán biểu thức, đặc biệt là biểu thức hậu tố. Các thuật toán tìm kiếm, duyệt và đệ quy cũng tận dụng stack để lưu trữ trạng thái và thực hiện quay lui khi cần thiết. Ngoài ra, stack còn được ứng dụng trong nhiều tính năng phổ biến như hoàn tác, làm lại và duyệt web.

Provide all the answers to the above questions in your submission ocument.

Practical - Stacks of Stacks ! (please re-design and explain how it work)

6. Start Logisim and open a new canvas
7. Review the lecture slides on building a stack at the top of this lab sheet. We are going to build a 5-bit deep, 1-bit wide stack.
8. Start by building a simple shift register that moves bits from one flip flop to the next each clock pulse. For this you will need a “Data In” pin which sets the next bit to be pushed to the stack, and a clock to invoke the shifting.



D_in: dữ liệu đầu vào (0 hoặc 1)

Q1, Q2, Q3, Q4, Q5: đại diện cho trạng thái của các flip-flops từ trái sang phải

Clk: số lần xung clock đã xảy ra

Thanh ghi dịch chuyển (Shift Register)
- Chức năng: Chuyển dữ liệu qua các Flip-Flop theo một hướng nhất định mỗi khi có xung nhịp (clock).
- Cách hoạt động:

+ Đầu vào (Data In): Giá trị dữ liệu được đưa vào Flip-Flop đầu tiên.

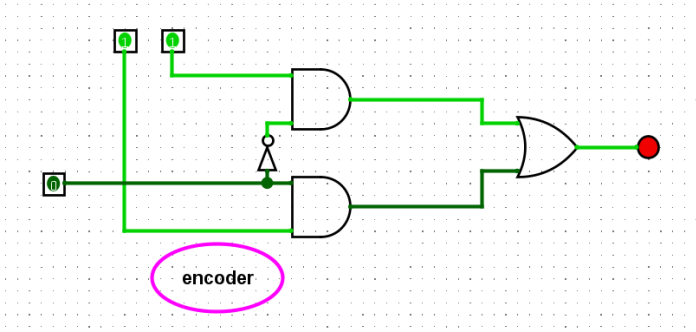
+ Xung nhịp (Clock): Mỗi xung nhịp sẽ di chuyển dữ liệu từ Flip-Flop hiện tại sang Flip-Flop tiếp theo trong chuỗi.

+ Đầu ra (Output): Dữ liệu từ Flip-Flop cuối cùng có thể được truy xuất.
Ứng dụng: Tạo một ngăn xếp đơn giản, nơi dữ liệu được thêm vào theo thứ tự và có thể được lấy ra theo phương pháp LIFO (Last In, First Out).

Bảng chân trị:

CLK	D_in	Q1	Q2	Q3	Q4	Q5
0	1	1	0	0	0	0
1	0	0	1	0	0	0
2	1	1	0	1	0	0
3	0	0	1	0	1	0
4	1	1	0	1	0	1

9. For your shift register to work as a stack, it needs to be bi-directional. This means the in- put to any Flip Flop could come from two places - the left or the right. In lectures we dis- cussed a simple “encoder” circuit that selects which of two data inputs is allowed through, based on a third selection bit. Design the logic for this 2-bit encoder, and demonstrate it to your lab demonstrator.



- Chức năng:

Mạch mã hóa 2 bit này có nhiệm vụ chọn một trong hai đầu vào dữ liệu để đưa vào flip-flop, dựa trên giá trị của bit chọn. Điều này cho phép thanh ghi dịch hoạt động hai chiều, cần thiết để thực hiện các thao tác push và pop của stack.

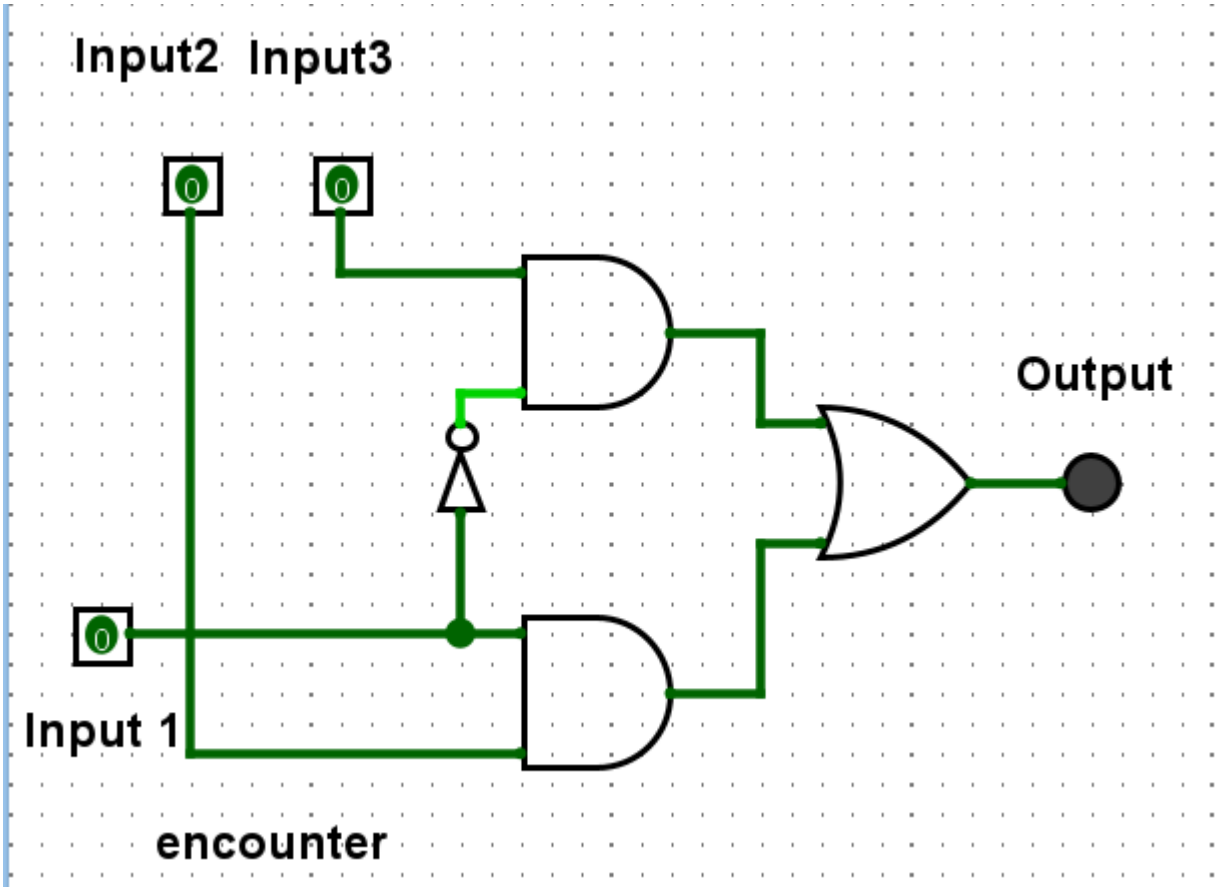
- Cách hoạt động:

Mạch encoder này sẽ có 2 đầu vào dữ liệu (Input 1 và Input 2) và một bit chọn (Select).

Dựa trên giá trị của bit chọn, mạch sẽ quyết định đầu vào nào được đưa ra output:

Select = 0: Input 1 được chọn, cho phép dịch chuyển sang phải (push vào stack).

Select = 1: Input 2 được chọn, cho phép dịch chuyển sang trái (pop ra khỏi stack).



Input 1	Input 2	Input 3	Not(Input 1)	And1(Input 3 & Not Input 1)	And2(Input 2 & Input 1)	Output
0	0	0	1	0	0	0
0	0	1	1	1	0	1
0	1	0	1	0	0	0
0	0	1	1	1	0	1
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	0	0	1	1
1	1	1	0	0	1	1

- Ứng dụng:

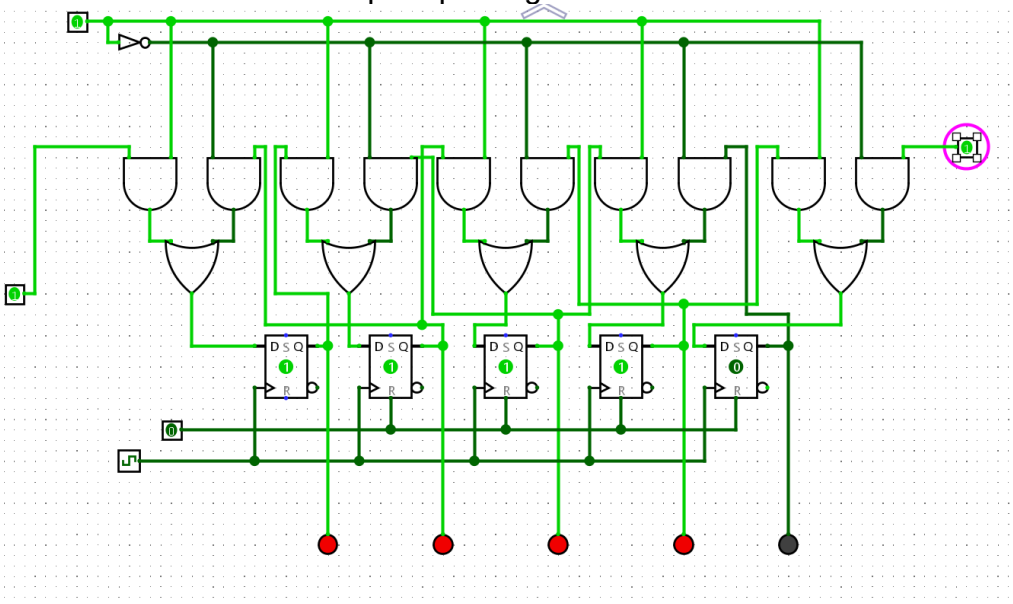
Chọn chip nhớ: Trong hệ thống có nhiều chip nhớ, decoder 3-to-8 có thể được sử dụng để chọn một trong 8 chip dựa trên địa chỉ bộ nhớ.

Điều khiển hiển thị: Decoder có thể được sử dụng để điều khiển các đoạn của màn hình 7 đoạn, hiển thị các số từ 0 đến 9.

Mạch giải mã địa chỉ: Trong vi xử lý, decoder được sử dụng để giải mã địa chỉ bộ nhớ hoặc thiết bị ngoại vi.

Chuyển đổi mã: Decoder có thể được sử dụng để chuyển đổi từ mã nhị phân sang các loại mã khác, ví dụ như mã Gray.

10. Now incorporate your encoder above to allow bi-directional shifting of your stack. Your stack should:
    10.1. push and pop bits onto and off the stack, using clock pulses and a direction toggle switch
    10.2. show the state of each Flip Flop using LEDs.



- Sơ đồ mô tả một hệ thống Bi-Directional Stack (Stack hai chiều). Một Bi-Directional Stack cho phép thêm (push) và xóa (pop) các phần tử từ cả hai đầu của stack, cung cấp sự linh hoạt hơn so với một stack truyền thống. Trong sơ đồ này, có 5 stack được kết nối với nhau.

- Cách hoạt động:

+ Push Mode: Dữ liệu từ Data In được đẩy vào Flip-Flop đầu tiên và dịch chuyển dần xuống qua các Flip-Flop khác.

+ Pop Mode: Dữ liệu từ Flip-Flop cuối cùng được lấy ra và các giá trị còn lại dịch chuyển ngược lên.

+ Direction Toggle Switch: Quyết định hướng dịch chuyển của dữ liệu (push hoặc

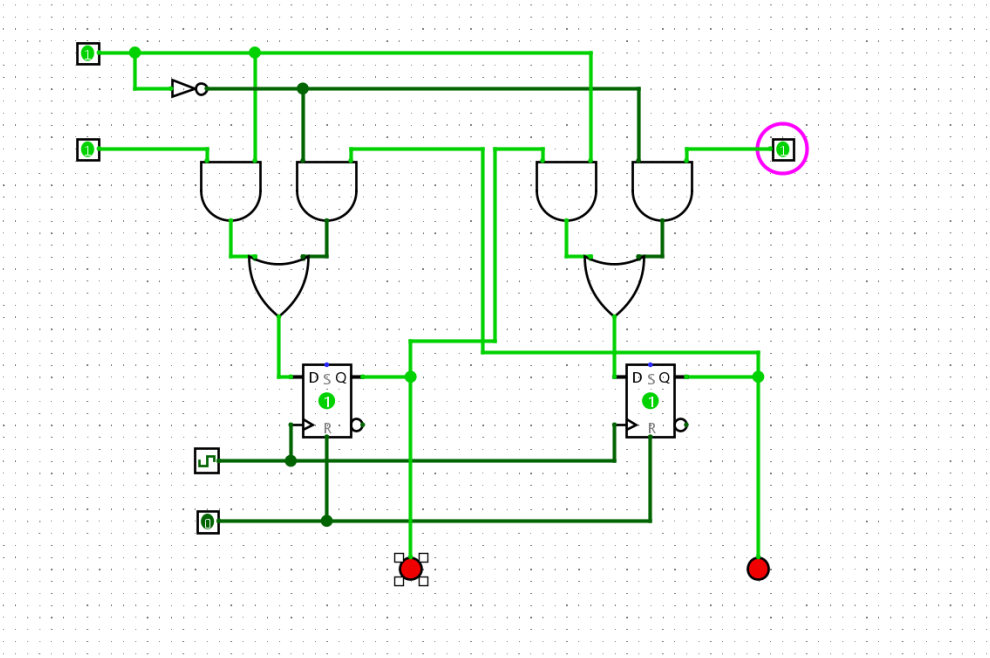
pop).

- Ứng dụng: Tăng tính linh hoạt của stack, có thể được sử dụng để lưu trữ lịch sử các thao tác trong ứng dụng, cho phép người dùng thực hiện các thao tác undo và redo.
- Bảng chân trị:

Clock	Input	Direction	FF1	FF2	FF3	FF4	FF5	Led State
0	X	X	0	0	0	0	0	Off
1	1	1	1	0	0	0	0	On
2	0	1	0	1	0	0	0	On
3	1	1	1	0	1	0	0	On
4	X	0	0	1	0	1	0	On

Export your circuit as an image and include it in your submission document. Demon- strate your working stack to your lab demonstrator.

11. Modify your stack so that it has the option to read out its contents ***in parallel*** to a sepa- rate register of D Flip Flops. This should only occur when a “stack dump” toggle switch (i.e., pin) is enabled. When the toggle is disabled, the register of D Flip Flops should re- tain the last state read in (and should have LEDs connected to each Flip Flop out showing its state).



- Chức năng:

Mạch này có chức năng sao chép dữ liệu từ thanh ghi dịch (hoạt động như stack) sang một thanh ghi D flip-flop khác khi nhận được tín hiệu "stack dump". Khi tín hiệu này

tắt, thanh ghi D flip-flop sẽ giữ nguyên dữ liệu cuối cùng được đọc.

- Cách hoạt động:

1. Lưu trữ dữ liệu:

+ Dữ liệu được đẩy vào stack (push) bằng cách dịch chuyển bit vào thanh ghi dịch từ một đầu.

+ Dữ liệu được lấy ra khỏi stack (pop) bằng cách dịch chuyển bit ra khỏi thanh ghi dịch từ đầu kia.

+ Hướng dịch chuyển (trái hoặc phải) được điều khiển bởi một mạch encoder (mạch mã hóa) dựa trên giá trị của bit chọn.

2. Đọc dữ liệu:

+ Khi công tắc "stack dump" được bật, dữ liệu từ mỗi flip-flop trong thanh ghi dịch được truyền trực tiếp sang đầu vào D tương ứng của thanh ghi D flip-flop.

+ Các flip-flop trong thanh ghi D flip-flop sẽ lưu trữ giá trị này ở cạnh lên hoặc cạnh xuống của xung clock.

3. Giữ dữ liệu:

+ Khi "stack dump" tắt, đầu vào D của các flip-flop trong thanh ghi D flip-flop không nhận dữ liệu mới.

+ Do đó, chúng sẽ giữ nguyên trạng thái (dữ liệu) cuối cùng được đọc từ stack.

4. Hiển thị dữ liệu:

+ Mỗi flip-flop trong thanh ghi D flip-flop được kết nối với một đèn LED.

+ Đèn LED sẽ sáng khi flip-flop ở trạng thái 1 và tắt khi flip-flop ở trạng thái 0, hiển thị trực quan dữ liệu được lưu trữ.

- Bảng chân trị:

Clock	Input	Direction	Stack Dump	FF1	FF2	FF3	FF4	FF5	Output
0	x	x	0	0	0	0	0	0	off
1	1	1	0	1	0	0	0	0	on
2	0	1	0	0	1	0	0	0	on
3	1	1	0	1	0	1	0	0	on
4	x	0	1	1	0	1	0	0	on
5	x	x	0	1	0	1	0	0	on

- Ứng dụng:

+ Gỡ lỗi: Kiểm tra nội dung của stack trong quá trình phát triển phần mềm hoặc phần cứng.

+ Giám sát hệ thống: Theo dõi dữ liệu quan trọng được lưu trữ trong stack.

+ Lưu trữ tạm thời: Lưu trữ tạm thời dữ liệu trước khi xử lý hoặc truyền đi.

+ Hệ thống nhúng**:** Ứng dụng trong các hệ thống nhúng với tài nguyên hạn chế, nơi cần kiểm tra nội dung stack một cách hiệu quả.

Export your circuit as an image and include it in your submission document. Demon- strate your working stack to your lab demonstrator.