

CÂU HỎI THI TRẮC NGHIỆM

Tên học phần: Kỹ Thuật Vi Xử Lý
Số tín chỉ: 03

Mã học phần: ELE1317

Thứ tự câu hỏi	Nội dung câu hỏi	Đáp án	Phương án trả lời				Điểm
			A	B	C	D	
1	Chức năng chính của CPU là gì?		Lưu trữ dữ liệu	Hiển thị dữ liệu	Xử lý dữ liệu	Gửi dữ liệu qua mạng	1
2	Khối nào của CPU chịu trách nhiệm thực hiện các phép tính cơ bản như cộng, trừ, nhân, chia?		ALU (Arithmetic Logic Unit)	RAM (Random Access Memory)	ROM (Read-Only Memory)	CU (Control Unit)	1
3	Thành phần nào trong CPU chịu trách nhiệm điều khiển hoạt động của máy tính và lấy dữ liệu từ bộ nhớ?		ALU (Arithmetic Logic Unit)	RAM (Random Access Memory)	ROM (Read-Only Memory)	CU (Control Unit)	1
4	Trong kiến trúc CISC, một lệnh có thể thực hiện nhiều tác vụ khác nhau bao gồm:		Các phép tính toán	Truy cập bộ nhớ	Kiểm soát luồng chương trình (Chuyển tới địa chỉ khác)	Cả ba câu kia đều đúng	1
5	Kiến trúc RISC tập trung vào điều gì?		Tăng đa nhiệm	Tối ưu hóa hiệu suất xử lý	Giảm độ phức tạp của lệnh	Điều khiển bộ nhớ	1
6	ROM và RAM đều là loại bộ nhớ trong máy tính. Tuy nhiên, điểm khác biệt		ROM lưu trữ dữ liệu dùng cho việc khởi động máy tính, trong	ROM lưu trữ dữ liệu dùng cho các chương trình ứng dụng, trong	ROM lưu trữ dữ liệu mà người dùng nhập vào, trong khi RAM	ROM lưu trữ dữ liệu lưu trữ tạm thời, trong khi RAM lưu trữ dữ	1

	quan trọng nhất giữa chúng là gì?		khi RAM lưu trữ dữ liệu làm việc trong quá trình hoạt động của máy tính	khi RAM lưu trữ dữ liệu hệ điều hành	lưu trữ dữ liệu hệ thống	liệu dùng cho việc khởi động máy tính	
7	Trong một hệ thống vi xử lý bộ nhớ ROM thường được sử dụng để:		Lưu các hằng biến của chương trình	Lưu các chương trình khởi động hệ thống và quản lý các thiết bị ngoại vi	Lưu các dữ liệu cần truy cập nhanh	Lưu các chương trình ứng dụng nạp từ ổ cứng	1
8	Trong một hệ thống vi xử lý bộ nhớ RAM thường được sử dụng để:		Lưu các hằng và các biến của chương trình	Lưu các chương trình khởi động hệ thống và quản lý các thiết bị ngoại vi	Lưu các dữ liệu cần truy cập nhanh	Lưu các chương trình ứng dụng nạp từ ổ cứng	1
9	BUS địa chỉ (Address Bus) trong hệ thống vi xử lý được sử dụng để làm gì?		Truyền dữ liệu giữa CPU với bộ nhớ và cổng vào ra	Điều khiển các chế độ hoạt động của hệ thống	Định vị vị trí sẽ truyền dữ liệu với CPU	Cả ba câu kia đều đúng	1
10	Trong một hệ thống vi xử lý BUS địa chỉ là các tín hiệu:		Ngõ vào	Ngõ ra	Hai chiều	Có tín hiệu chỉ vào, chỉ ra, hai chiều	1
11	Trong một hệ thống vi xử lý BUS dữ liệu là các tín hiệu:		Ngõ vào	Ngõ ra	Hai chiều	Có tín hiệu chỉ vào, chỉ ra, hai chiều	1
12	Trong một hệ thống vi xử lý BUS điều khiển là các tín hiệu:		Ngõ vào	Ngõ ra	Hai chiều	Có tín hiệu chỉ vào, tín hiệu chỉ ra, và tín hiệu hai chiều	1
13	BUS điều khiển (Control Bus) trong hệ thống vi xử		Truyền dữ liệu giữa CPU với bộ	Điều khiển các chế độ hoạt động của hệ thống	Định vị vị trí sẽ truyền dữ liệu với CPU	Cả ba câu kia đều đúng	2

	lý được sử dụng để làm gì?		nhớ và cổng vào ra				
14	Khác nhau cơ bản giữa vi xử lý và vi điều khiển là:		Hệ thống vi xử lý thường sử dụng cho các ứng dụng lớn hoặc đa năng. Còn vi điều khiển thường sử dụng cho các ứng dụng chuyên dụng	Vi xử lý không thể hoạt động một mình, nó cần kết nối với bộ nhớ và vào ra để tạo thành hệ thống vi xử lý. Còn vi điều khiển là một hệ thống vi xử lý tích hợp trong một IC	Vi xử lý không tích hợp các chức năng ngoại vi, như GPIO, UART, SPI, v.v. Còn vi điều khiển thường tích hợp các thành phần trên	Cả ba câu kia đều đúng	2
15	ALU thường thực hiện các phép xử lý dữ liệu nào?		Các phép so sánh	Các phép tính số học	Các phép tính logic	Cả ba câu kia đều đúng	2
16	Tín hiệu chọn một phép tính thực hiện trong ALU được cung cấp từ:		Thanh ghi PC	CU	Vào ra	Bộ nhớ	2
17	Khi nào thì cổng vào ra được mở để CPU truyền dữ liệu tới một ngoại vi?		Khi CPU cấp đúng địa chỉ của cổng vào ra và tín hiệu điều khiển tích cực	Khi thiết bị ngoại vi tác động mở cổng vào ra	Khi vào ra nhận được dữ liệu cấp tới từ thiết bị ngoại vi	Cả ba câu kia đều đúng	2
18	CPU có 16 đường địa chỉ thì sẽ quản lý được bao nhiêu vị trí bộ nhớ?		65536	16384	8192	32768	2
19	Dung lượng bộ nhớ mà CPU quản lý sẽ là bao nhiêu nếu nó có 20 đường		2MB	1MB	4MB	8MB	2

	địa chỉ và 8 đường dữ liệu?						
20	Điều gì xảy ra khi thanh ghi bộ đếm chương trình PC trở vào vùng nhớ dữ liệu?		Người lập trình sẽ được thông báo bằng một cơ chế đặc biệt	CPU sẽ tự động điều chỉnh giá trị của PC để quay về vùng nhớ chứa mã lệnh	Sẽ xảy ra ngoại lệ	CPU sẽ thực hiện các lệnh mà người lập trình không kiểm soát được	2
21	Nếu CPU sử dụng 8 bit địa chỉ thì nó có thể quản lý được bao nhiêu vị trí vào ra?		256	128	64	32	2
22	Các bộ ADC và DAC trong hệ thống vi xử lý thực hiện chức năng gì?		Biến đổi tín hiệu tương tự từ CPU thành tín hiệu số cung cấp cho ngoại vi, và ngược lại biến đổi tín hiệu tương tự từ CPU thành tín hiệu số cung cấp ra bên ngoài ngoại vi	Giao tiếp tín hiệu tương tự giữa các ngoại vi của hệ thống với nhau	Biến đổi tín hiệu tương tự từ ngoại vi thành tín hiệu số cung cấp cho CPU, và ngược lại biến đổi tín hiệu số từ CPU thành tín hiệu tương tự cung cấp ra bên ngoài ngoại vi	Cả ba câu kia đều đúng	2
23	PWM thường được sử dụng trong các ứng dụng điều khiển động cơ để làm gì?		Tăng tốc độ động cơ	Giảm tốc độ động cơ	Đảo chiều quay của động cơ	Câu ##A và ##B đều đúng	2
24	PWM là viết tắt của gì?		Pulse Width Modulation	Programmed Wave Manipulation	Power Width Modulation	Pulse Waveform Manipulation	2
25	PWM có thể được tạo ra trên một vi điều khiển hoặc hệ thống vi xử lý		Điện áp đầu vào	Điện áp nguồn	Timer/Counter	Điều khiển từ xa	2

	bằng cách sử dụng khối chức năng nào?						
26	Khi có các thanh ghi chương trình sẽ thực hiện nhanh hơn do:		Thanh ghi chưa các dữ liệu trung gian của chương trình, giảm bớt các chu kỳ đọc ghi bộ nhớ	Thanh ghi được sử dụng làm nơi chứa các kết quả cuối cùng của chương trình	Thanh ghi sử dụng làm nơi chứa các giá trị biến cung cấp từ các thiết bị vào	Cả ba câu kia đều đúng	2
27	CPU sử dụng tín hiệu của BUS nào để thông báo cho vào ra biết cần cấp hay nhận dữ liệu?		Địa chỉ	Dữ liệu	Điều khiển	Cả ba câu kia đều đúng	2
28	Các chế độ địa chỉ “Addressing mode” là:		Các cách mà một lệnh xử lý các dữ liệu	Các cách mà một lệnh hoặc một hướng dẫn lấy địa chỉ hoặc dữ liệu	Là các vị trí của dữ liệu và mã lệnh trong bộ nhớ	Cả ba câu kia đều đúng	2
29	Các CPU CISC sẽ có các lợi điểm gì so với các CPU RISC?		CISC thường hỗ trợ nhiều chức năng phức tạp trong lệnh, điều này giúp đơn giản hóa quá trình lập trình cho các ứng dụng phức tạp như hệ điều hành và phần mềm ứng dụng	CISC hỗ trợ các kiểu dữ liệu đơn giản nên dễ dàng trong việc lập trình hơn	CISC tích hợp nhiều chức năng phức tạp trong một lệnh nên quá trình thực hiện chương trình sẽ nhanh chóng hơn	Cả ba câu kia đều đúng	2
30	Cấu trúc đường ống (Pipeline) là gì?		Là việc thực hiện đồng thời nhiều lệnh tại	Là việc thực hiện tuần tự từng lệnh, lệnh này	Là việc thực hiện các lệnh một cấu	Cả ba câu kia đều đúng	2

			một thời điểm, mỗi lệnh thực hiện một công đoạn khác nhau	thực hiện xong mới tới lệnh kế tiếp	trú ồng trong vi xử lý		
31	Chương trình trong các hệ thống vi xử lý là gì?		Là tập hợp các lệnh sắp xếp theo một thuật toán nhất định để xử lý một công việc hay một bài toán nhất định	Là tập lệnh của vi xử lý do nhà sản xuất công bố	Là tập hợp các lệnh được người sử dụng sắp xếp một cách ngẫu nhiên	Cả ba câu kia đều đúng	2
32	Các chương trình chứa trong bộ nhớ để CPU thực hiện là các chương trình dưới dạng:		Các chương trình bằng ngôn ngữ bất kỳ	Các chương trình ngôn ngữ cấp cao	Các chương trình mã máy (Machine language)	Các chương trình hợp ngữ (Assembly)	2
33	Để đọc dữ liệu từ bộ nhớ, CPU cần thực hiện các công việc nào?		CPU cấp địa chỉ, cấp tín hiệu điều khiển chọn bộ nhớ, cấp tín hiệu yêu cầu đọc bộ nhớ và nhận dữ liệu từ data bus vào	CPU nhận địa chỉ từ bộ nhớ, cấp tín hiệu điều khiển chọn bộ nhớ, cấp tín hiệu yêu cầu đọc bộ nhớ và nhận dữ liệu từ data bus vào	CPU cấp địa chỉ, cấp tín hiệu điều khiển chọn bộ nhớ, cấp tín hiệu yêu cầu đọc bộ nhớ và cấp dữ liệu ra data bus vào	CPU cấp địa chỉ, cấp tín hiệu điều khiển chọn bộ nhớ, nhận tín hiệu cho phép đọc bộ nhớ và nhận dữ liệu từ data bus vào	3
34	Để ghi dữ liệu ra bộ nhớ, CPU cần thực hiện các công việc nào?		CPU cấp địa chỉ, cấp tín hiệu điều khiển chọn bộ nhớ, cấp tín hiệu yêu cầu ghi bộ nhớ và nhận dữ	CPU nhận địa chỉ từ bộ nhớ, cấp tín hiệu điều khiển chọn bộ nhớ, cấp tín hiệu yêu cầu ghi bộ nhớ và nhận dữ	CPU cấp địa chỉ, cấp tín hiệu điều khiển chọn bộ nhớ, cấp tín hiệu yêu cầu ghi bộ nhớ và cấp dữ liệu ra data bus	CPU cấp địa chỉ, cấp tín hiệu điều khiển chọn bộ nhớ, nhận tín hiệu cho phép ghi bộ nhớ và	3

			liệu từ data bus vào	liệu từ data bus vào		cấp dữ liệu ra data bus	
35	Để đọc dữ liệu từ vào ra, CPU cần thực hiện các công việc nào?		CPU cấp địa chỉ, cấp tín hiệu điều khiển chọn vào ra, cấp tín hiệu yêu cầu đọc vào ra và nhận dữ liệu từ data bus vào	CPU nhận địa chỉ từ vào ra, cấp tín hiệu điều khiển chọn vào ra, cấp tín hiệu yêu cầu đọc vào ra và nhận dữ liệu từ data bus vào	CPU cấp địa chỉ, cấp tín hiệu điều khiển chọn vào ra, nhận tín hiệu cho phép đọc từ vào ra và nhận dữ liệu từ data bus vào	CPU cấp địa chỉ, cấp tín hiệu điều khiển chọn vào ra, cấp tín hiệu yêu cầu đọc vào ra và cấp dữ liệu ra data bus	3
36	Để ghi dữ liệu ra vào ra, CPU cần thực hiện các công việc nào?		CPU cấp địa chỉ, cấp tín hiệu điều khiển chọn vào ra, cấp tín hiệu yêu cầu ghi vào ra và nhận dữ liệu từ data bus vào	CPU nhận địa chỉ từ vào ra, cấp tín hiệu điều khiển chọn vào ra, cấp tín hiệu yêu cầu phép ghi vào ra và cấp dữ liệu ra data bus	CPU cấp địa chỉ, cấp tín hiệu điều khiển chọn vào ra, cấp tín hiệu yêu cầu phép ghi vào ra và cấp dữ liệu ra data bus	CPU cấp địa chỉ, cấp tín hiệu điều khiển chọn vào ra, nhận tín hiệu cho phép ghi từ vào ra và cấp dữ liệu ra data bus	3
37	Để đọc lệnh, CPU cần thực hiện các công việc nào?		CPU nhận địa chỉ vào PC, cấp tín hiệu chọn bộ nhớ, cấp tín hiệu đọc bộ nhớ và lấy mã lệnh từ data bus	CPU cấp địa chỉ từ PC, cấp tín hiệu chọn bộ nhớ, cấp tín hiệu đọc bộ nhớ và lấy mã lệnh từ data bus	CPU cấp địa chỉ từ PC, cấp tín hiệu chọn bộ nhớ, nhận tín hiệu đọc từ bộ nhớ và lấy mã lệnh từ data bus	Cả ba câu kia đều đúng	3
38	Khi thực hiện các lệnh với dữ liệu nằm trên các thanh ghi bên trong CPU, data bus của CPU sẽ ở trạng thái		Là các ngõ vào	Là các ngõ ra	Là các đường trở kháng cao	Mang các mức logic không xác định	3

39	Thực hiện thiết kế các hệ thống điều khiển bằng MCU sẽ đơn giản và mất ít thời gian hơn thực hiện thiết kế bằng hệ thống vi xử lý là do:		Mạch phân cứng đơn giản	Việc lập trình đơn giản hơn	Các công cụ lập trình và biên dịch phần mềm tiện dụng hơn	Cả ba câu kia đều đúng	3
40	Để xác định đã hết thời gian trễ hoặc đủ số sự kiện, Timer/Counter có thể:		Chờ điều kiện tràn khi đếm tăng	Chờ điều kiện so sánh bằng giá trị định trước	Chờ giá trị bộ đếm bằng 0 khi đếm xuống	Cả ba câu kia đều đúng	3
41	Trong cấu trúc một CPU truyền thống thanh ghi chứa (Acc) thường có chức năng gì khi ALU thực hiện các phép tính số học và logic?		Chứa kết quả của các phép tính	Giữ một toán hạng và kết quả của phép tính	Chứa trạng thái của các phép tính	Chứa các toán hạng của phép tính	1
42	ALU thực hiện các phép tính nhiều nhất có bao nhiêu toán hạng?		1	2	3	Tùy thuộc vào loại cấu trúc ALU khác nhau	1
43	Các toán hạng mà ALU thực hiện có thể nằm ở đâu trong hệ thống vi xử lý?		Trong các thanh ghi	Trong vùng nhớ dữ liệu	Trong vùng nhớ chương trình	Cả ba câu kia đều đúng	1
44	Chức năng chính của BIU là gì?		Quản lý giao tiếp giữa CPU với các thành phần bên ngoài hệ thống vi xử lý	Thực hiện phép tính số học	Điều khiển hoạt động của CPU	Thực hiện phép tính logic	1
45	Chức năng nào sau đây là đúng cho thanh ghi bộ đếm chương trình (PC) của CPU:		Đếm số lệnh mà CPU đã thực hiện	Giữ địa chỉ bộ nhớ để lấy các lệnh vào CPU	Đếm số dữ liệu của chương trình mà CPU thực hiện	Giữ mã lệnh cung cấp cho CU giải mã thực hiện	1

46	Hàng đợi lệnh (Instruction Queue) có chức năng:		Chứa các mã lệnh lấy vào từ bộ nhớ	Điều khiển việc chờ để lấy các lệnh từ bộ nhớ	Chứa các địa chỉ lấy lệnh của các lệnh tiếp theo trong chương trình	Cả ba câu kia đều đúng	1
47	Các thanh ghi địa chỉ bộ nhớ (MAR) có chức năng:		Chứa địa chỉ của đỉnh ngăn xếp	Chứa địa chỉ của lệnh sẽ được thực hiện	Chứa địa chỉ để truy cập tới các ô nhớ và vào ra	Chứa địa chỉ của các thanh ghi khác	1
48	Thanh ghi con trỏ ngăn xếp (SP) có các chức năng:		Cung cấp địa chỉ để đọc ghi ngăn xếp, thay đổi giá trị sau các lệnh truy cập ngăn xếp để sử dụng cho lần truy cập tiếp theo	Trỏ tới vị trí của ngăn xếp nằm ngoài thiết bị ngoại vi lưu trữ dữ liệu	Cung cấp địa chỉ để đọc ghi ngăn xếp, thay đổi dữ liệu lấy được từ ngăn xếp	Cả ba câu kia đều đúng	1
49	Thanh ghi chỉ số (Index Register) có chức năng:		Giữ giá trị độ dời địa chỉ trong các lệnh truy cập bộ nhớ theo chế độ chỉ số	Giữ địa chỉ cơ sở trong các lệnh truy cập bộ nhớ theo chế độ chỉ số	Giữ dữ liệu truy cập từ bộ nhớ trong các lệnh truy cập bộ nhớ theo chế độ chỉ số	Giữ độ dời để trỏ tới lệnh kế tiếp trong các lệnh truy cập bộ nhớ theo chế độ chỉ số	1
50	Thông thường các bit trong thanh ghi cờ bao gồm:		Các bit thông báo trạng thái các phép tính mà ALU thực hiện	Các bit thông báo trạng thái các chế độ hoạt động của CPU	Các bit điều khiển các chế độ hoạt động của CPU	Cả ba câu kia đều đúng	1
51	Điều khiển chương trình sẽ thay đổi không tuân theo cơ chế tuần tự trong các cơ chế:		Lệnh rẽ nhánh	Lệnh chương trình con	Phục vụ ngắt	Cả ba câu kia đều đúng	1
52	Trong lệnh rẽ nhánh tương đối, địa chỉ của		Được lấy từ đỉnh ngăn xếp	Giá trị hiện tại của PC cộng hoặc trừ một số	Là một giá trị được mã hóa trong lệnh	Là giá trị của thanh ghi cờ	1

	lệnh tiếp theo thực hiện là:			độ dài mã hóa trong lệnh			
53	Trạng thái ARM của ARM7TDMI các lệnh thực hiện với dữ liệu bao nhiêu bit?		8	16	32	64	1
54	ARM là viết tắt của gì?		Advanced Reduced Instruction Set Computer	Advanced Random Information System Core	Advanced RISC Machine	Advanced Complex Instruction Set Computer	1
55	Điểm mạnh chính của vi xử lý ARM là gì?		Hiệu suất cao	Tiêu thụ năng lượng thấp	Số lượng lệnh phong phú	Tương thích đa nền tảng	1
56	ARM7TDMI có bao nhiêu chế độ hoạt động?		5	6	7	8	1
57	Thanh ghi PC (Program Counter) trong ARM7 được sử dụng để làm gì?		Lưu trữ dữ liệu chương trình	Đếm số lần thực hiện lệnh	Lưu trữ địa chỉ của lệnh tiếp theo cần thực hiện	Lưu trữ địa chỉ của bộ nhớ RAM	1
58	Thanh ghi CPSR (Current Program Status Register) trong ARM7 được sử dụng để làm gì?		Lưu trạng thái hiện tại của chương trình	Lưu trữ dữ liệu người dùng	Lưu trữ dữ liệu tạm thời	Lưu trữ địa chỉ của bộ nhớ ROM	1
59	ARM7TDMI có bao nhiêu tập lệnh khác nhau?		2	4	6	8	1
60	Trong kiến trúc ARM7, thanh ghi LR (Link Register) được sử dụng để làm gì?		Lưu trạng thái hiện tại của chương trình	Lưu trữ dữ liệu người dùng	Lưu trữ địa chỉ của lệnh quay lại sau một lệnh nhảy (branch)	Lưu trữ địa chỉ của bộ nhớ RAM	1
61	Trong kiến trúc ARM7, thanh ghi SP (Stack Pointer) được sử dụng để làm gì?		Lưu trạng thái hiện tại của chương trình	Lưu trữ dữ liệu người dùng	Quản lý ngăn xếp (stack) của chương trình	Lưu trữ địa chỉ của bộ nhớ ROM	1

62	ARM7TDMI có bao nhiêu thanh ghi?		35	36	37	38	1
63	Bit T trong thanh ghi CPSR của ARM7TDMI sử dụng để:		Thay đổi chế độ hoạt động (User, FIQ, IRQ, ...)	Là bit bẫy (Trap) khi thực hiện chương trình	Là bit báo trạng thái hoạt động của CPU	Cả ba câu kia đều đúng	1
64	Các chế độ (Mode) hoạt động của ARM là		User, System, Supervisor	FIQ, IRQ, reset	Abort, Undefined	Cả ba câu kia đều đúng	1
65	Khi giải mã một lệnh mà ALU thực hiện, tín hiệu CU cung cấp để xác định những gì?		Khởi mạch phần cứng thực hiện phép toán; Vị trí cung cấp dữ liệu cho phép toán	Vị trí ghi các bit trạng thái của phép toán thực hiện; Vị trí chứa kết quả phép toán	Vị trí lấy lệnh tiếp theo	Cả ##A và ##B đều đúng	2
66	Khi sử dụng thanh ghi chứa (Acc) sẽ có lợi ích gì?		Chứa được nhiều dữ liệu hơn	Mã lệnh số học logic sẽ ngắn do không phải mã hóa một toán hạng và nơi chứa kết quả của lệnh	Giảm việc đọc ghi bộ nhớ để truy cập các toán hạng	Cả ba câu kia đều đúng	2
67	ALU cung cấp các cờ nào tới thanh ghi cờ?		Các cờ điều khiển các chế độ xử lý dữ liệu khác nhau của CPU	Các cờ điều khiển các chế độ giao tiếp với bên ngoài của CPU	Các cờ thông báo trạng thái kết quả của các phép tính mà ALU thực hiện	Cả ba câu kia đều đúng	2
68	Sau khi thực hiện xong một lệnh giá trị của thanh ghi PC sẽ thay đổi như thế nào?		Giữ nguyên giá trị hiện tại đang trở tới lệnh viết kế tiếp trong chương trình	Cộng trừ đi một giá trị lưu trong mã lệnh để trở tới một vị trí lấy lệnh mới	Nạp một giá trị mới lưu trong mã lệnh để trở tới một vị trí lấy lệnh mới	Cả ba câu kia đều đúng	2
69	Hàng đợi lệnh là một cấu trúc bộ nhớ hoạt động theo nguyên tắc:		FIFO	FILO	LIFO	FOLI	2

70	Bằng nguyên tắc nào mà trong ngăn xếp dữ liệu nào cất vào sau cùng sẽ được lấy ra trước tiên (LIFO)?		Do thanh ghi SP có cơ chế chứa được nhiều dữ liệu một cách tuần tự	Do mỗi lần cất vào ngăn xếp dữ liệu sẽ được tự động đẩy xuống địa chỉ thấp, và khi lấy một dữ liệu ra khỏi ngăn xếp dữ liệu trong ngăn xếp sẽ tự động được đẩy lên địa chỉ cao hơn	Do khi cất dữ liệu vào ngăn xếp trước hết SP sẽ giảm và dữ liệu sẽ được cất vào địa chỉ SP giữ, và khi lấy ra từ ngăn xếp dữ liệu sẽ lấy từ địa chỉ SP (là dữ liệu cất vào sau cùng), sau đó SP mới tự động tăng để trở tới dữ liệu cất vào trước đó	Cả ba câu kia đều đúng	2
71	Ngăn xếp có các chức năng gì?		Cả ba câu kia đều đúng	Lưu các cờ và các trạng thái hoạt động hiện hành của CPU khi CPU thực hiện các chương trình con hoặc các chương trình phục vụ ngắt	Lưu trữ các dữ liệu trung gian còn sử dụng trong các thanh ghi để có thể sử dụng các thanh ghi cho các dữ liệu khác	Lưu địa chỉ quay về khi CPU thực hiện các chương trình con hoặc các chương trình phục vụ ngắt	2
72	Cho biết sự khác nhau khi truy cập bộ nhớ bằng các lệnh di chuyển dữ liệu sử dụng thanh ghi MAR cấp đại chỉ, và bằng các lệnh ngăn xếp sử dụng thanh ghi SP cấp địa chỉ?		Sau khi truy cập bộ nhớ thanh ghi MAR sẽ tự động thay đổi giá trị, còn thanh ghi SP có giá trị không đổi	Sau khi truy cập bộ nhớ thanh ghi MAR có giá trị không đổi còn thanh ghi SP sẽ tự động thay đổi giá trị	Sau khi truy cập bộ nhớ bằng MAR dữ liệu trong bộ nhớ không đổi còn thanh ghi SP dữ liệu trong bộ nhớ sẽ tự động thay đổi	Sau khi truy cập bộ nhớ bằng MAR dữ liệu trong bộ nhớ thay đổi còn thanh ghi SP dữ liệu trong bộ nhớ không đổi	2

73	Chương trình con phục vụ ngắt INT được thực hiện khi:		Cả ba câu kia đều đúng	Cờ cho phép ngắt được xóa và có tín hiệu yêu cầu ngắt cấp tới CPU	Khi che ngắt và có tín hiệu yêu cầu ngắt cấp tới CPU	Cờ cho phép ngắt được lập và có tín hiệu yêu cầu ngắt cấp tới CPU	2
74	Các lệnh rẽ nhánh có điều kiện sẽ sử dụng điều kiện chứa tại:		Các bit trạng thái của thanh ghi cờ	Các bit điều khiển chế độ của thanh ghi cờ	Toàn bộ các bit của thanh ghi cờ	Các bit chứa trong mã lệnh	2
75	Cơ chế chuyển điều khiển chương trình bằng lệnh rẽ nhánh và lệnh gọi chương trình con có gì khác nhau?		Lệnh rẽ nhánh chuyển điều khiển chương trình đi nhiều vị trí	Lệnh rẽ nhánh không lưu địa chỉ quay về và các trạng thái hoạt động hiện tại của CPU	Lệnh gọi chương trình con không lưu địa chỉ quay về và các trạng thái hoạt động hiện tại của CPU	Lệnh gọi chương trình con chuyển điều khiển chương trình đi nhiều vị trí	2
76	Cho biết sự khác nhau giữa cơ chế chuyển điều khiển chương trình bằng ngắt cứng và bằng chương trình con		Chuyển điều khiển bằng chương trình con không lưu địa chỉ quay về và các trạng thái hoạt động hiện tại của CPU	Chuyển điều khiển bằng ngắt cứng không lưu địa chỉ quay về và các trạng thái hoạt động hiện tại của CPU	Chuyển điều khiển bằng ngắt cứng được thực hiện bằng lệnh còn chuyển điều khiển bằng chương trình con tác được động bằng tín hiệu điện phân cứng	Chuyển điều khiển bằng ngắt cứng được tác động bằng tín hiệu điện phân cứng còn chuyển điều khiển bằng chương trình con được thực hiện bằng lệnh	2
77	Các yếu tố khác nhau giữa các CPU RISC và CISC là:		Số lượng lệnh; Thời gian thực hiện các loại lệnh	Địa chỉ lệnh	Số lượng thanh ghi	Cả ba câu kia đều đúng	2
78	Các lệnh của các bộ vi xử lý RISC thường có độ dài bằng nhau nhằm mục đích gì?		Dễ dàng phân đoạn trong cấu trúc đường ống	Tối ưu hóa việc sử dụng bộ đệm lệnh (Cache)	Cấu trúc giải mã lệnh đơn giản; Giảm độ phức tạp của việc xử lý lệnh	Cả ba câu kia đều đúng	2

79	Các loại bộ nhớ thường có trong một vi điều khiển (MCU) bao gồm:		Vùng nhớ ROM chứa chương trình	Vùng nhớ SRAM sử dụng cho các hằng và biến	Vùng nhớ ROM Flash sử dụng cho các dữ liệu không mất khi mất nguồn	Cả ba câu kia đều đúng	2
80	Khi kết quả phép tính bằng không thì:		CF =1	ZF = 1	NF =1	VF = 1	2
81	Các vi điều khiển thường có các loại vào ra gì?		UART	GPIO	SPI, I2C	Cả ba câu kia đều đúng	2
82	Các vi xử lý ARM được sử dụng trong hầu hết các thiết bị điện tử di động do đặc tính gì?		Hiệu suất hoạt động cao	Cấu trúc nhỏ gọn	Tiêu thụ nguồn thấp	Thực hiện được các lệnh phức tạp	2
83	Các chế độ tiết kiệm năng lượng của một vi điều khiển bao gồm:		Sleep	Idle	Power down	Cả ba câu kia đều đúng	2
84	MMU có chức năng gì trong hệ thống vi xử lý?		Quản lý bộ nhớ ảo	Quản lý bộ nhớ dữ liệu	Quản lý bộ nhớ chương trình	Quản lý tất cả các loại bộ nhớ trong hệ thống	2
85	Bit I trong thanh ghi CPSR của ARM7TDMI sử dụng để:		Thực hiện các lệnh với số nguyên (Integer)	Cho phép ngắt	Thực hiện các thao tác dữ liệu bên trong CPU	Cả ba câu kia đều đúng	2
86	Các ngoại lệ SWI và Reset sẽ làm ARM chuyển qua hoạt động ở chế độ:		User	Supervisor	System	Abort	2
87	IRQ và FIQ là các ngắt xảy ra khi:		Có tín hiệu phản ứng cấp tới các chân yêu cầu ngắt tương ứng	Thực hiện các lệnh phần mềm tương ứng	Khi chuyển qua chế độ hoạt động bằng cách thay đổi các bit M trong CPSR	Cả ba câu kia đều đúng	2

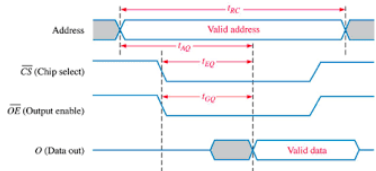
88	Các bit cờ điều kiện trong thanh ghi trạng thái chương trình CPSR (Current Program Status Register) sử dụng để:		Làm điều kiện rẽ nhánh chương trình, hoặc làm điều kiện để thực hiện lệnh	Thông báo trạng thái các lệnh số học và logic	Thông báo trạng thái các lệnh di chuyển dữ liệu	Cả ba câu kia đều đúng	2
89	Bit N trong thanh ghi CPSR của ARM7TDMI sẽ được lập khi:		Kết quả nhỏ hơn không	Thực hiện lệnh NOT	Khi không thực hiện lệnh tính toán	Cả ba câu kia đều đúng	2
90	Bit C trong thanh ghi CPSR của ARM7TDMI sẽ được lập khi:		Kết quả lệnh cộng tràn khỏi thanh ghi	Khi kết quả lệnh trừ có mượn	Khi kết quả lệnh nhân cần mở rộng khỏi thanh ghi	Cả ba câu kia đều đúng	2
91	Bit V trong thanh ghi CPSR của ARM7TDMI sẽ được lập khi:		Kết quả lệnh cộng tràn khỏi thanh ghi	Kết quả các phép tính có dấu đổi dấu không mong muốn	Khi kết quả lệnh nhân cần mở rộng khỏi thanh ghi	Cả ba câu kia đều đúng	2
92	Trong chế độ giám sát (SVC) của ARM7 có thể:		Chạy chương trình thiết lập hệ thống	Truy cập dữ liệu của hệ thống	Truy cập dữ liệu của người sử dụng	Cả ba câu kia đều đúng	2
93	Chế độ nào của ARM không truy cập được bộ nhớ và vào ra		User	Supervisor	System	Abort	2
94	Chế độ User của ARM cần bắt đầu hoạt động sau:		Chế độ System	Chế độ Supervisor	Chế độ IRQ	Chế độ FIQ	2
95	Khi cần hiệu chỉnh các tham số của hệ thống có thể sử dụng chế độ nào của ARM?		User	IRQ	System	Abort	2
96	Chế độ FIQ của ARM thực hiện nhanh các ISR do:		Chép nhanh các thanh ghi chương trình chính	Không lưu các trạng thái và thanh ghi chương trình chính	Không thực hiện các ISR	Cả ba câu kia đều đúng	2

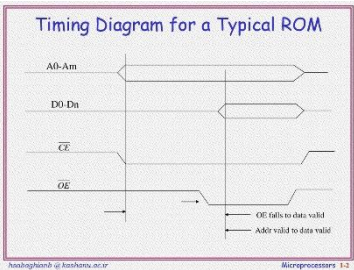
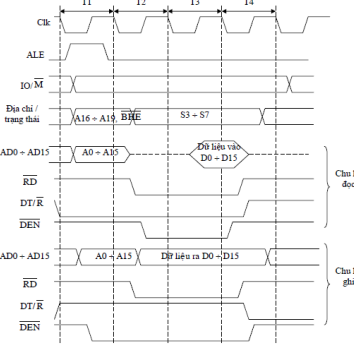
97	Chế độ Undefined sẽ xảy ra khi:		Thực hiện các chu kỳ không được định nghĩa	Xảy ra ngắt không định nghĩa	Thực hiện một lệnh không có trong tập lệnh	Xảy ra các lỗi truy cập bộ nhớ vào ra	2
98	Chế độ Abort sẽ xảy ra khi:		Truy cập vị trí bộ nhớ bị lỗi	Truy cập dữ liệu hệ thống không được cho phép	Truy cập các ứng dụng của hệ thống không được cho phép	Cả ba câu kia đều đúng	2
99	Khi xảy ra các ngoại lệ (Exception) ARM sẽ thực hiện các lệnh:		Tại các vị trí cố định xác định trước trong bộ nhớ	Tại các vị trí cung cấp bởi bộ điều khiển ngắt	Tại các vị trí không được xác định trước	Không có vị trí xác định	2
100	Khi xảy ra ngoại lệ địa chỉ quay về sẽ được lưu trữ:		Ở định ngăn xếp	Ở thanh ghi đa năng	Các thanh ghi LR tương ứng	Ở các vị trí cố định trong bộ nhớ	2
101	Công việc CPU ARM thực hiện khi xảy ra ngoại lệ:		Lưu địa chỉ quay về	Lưu thanh ghi CPSR vào SPSR	Đổi chế độ hoạt động, lập cờ cấm ngắt	Cả ba câu kia đều đúng	2
102	Công việc CPU ARM thực hiện khi kết thúc một ngoại lệ:		Dùng giá trị của LR để phục hồi giá trị của PC	Phục hồi giá trị của CPSR từ SPSR	Xóa các cờ cấm ngắt	Cả ba câu kia đều đúng	2
103	Khi đang thực hiện một ISR ngoại lệ, cơ cấm ngắt của ARM được lập để:		Chương trình ISR không dừng để chuyển qua một ISR khác	Làm điều kiện để kết thúc ISR khi nó thực hiện xong	Chuyển qua một ISR khác khi có yêu cầu	Cả ba câu kia đều đúng	2
104	Khi quay về từ một ISR của một ngoại lệ, giá trị thanh ghi PC của ARM được phục hồi bằng cách:		Lấy giá trị của thanh ghi LR	Tính toán từ giá trị hiện tại của PC và LR	Lấy giá trị từ định ngăn xếp	Lấy giá trị từ thanh ghi SPSR	2
105	Nếu ALU có các khối mạch thực hiện các phép tính số học và logic bao gồm: Cộng, trừ, nhân,		8	3	4	7	3

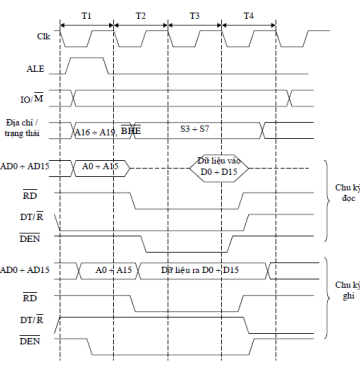
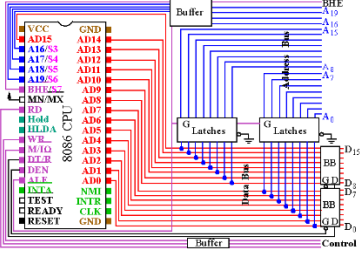
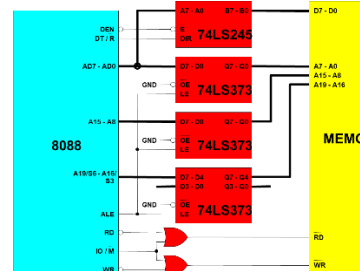
	chia, AND, OR, XOR, NOT thì nó cần nhận được ít nhất bao nhiêu tín hiệu điều khiển từ CU?						
106	Các khối chức năng số có thể có trong một BIU là:		Đệm hai chiều cho bus dữ liệu	Đệm một chiều cho bus địa chỉ	Mạch định thời cho các tín hiệu điều khiển	Cả ba câu kia đều đúng	3
107	Các khối chức năng số có thể có trong một Timer/Counter là:		Bộ đếm, mạch so sánh	Mạch giải mã, Mạch mã hóa	Mạch nhân, Mạch chia	Mạch cộng, Mạch trừ	3
108	Tại sao khi sử dụng thanh ghi địa chỉ bộ nhớ, độ dài mã lệnh truy cập bộ nhớ sẽ ngắn hơn?		Do CU không phải tính toán để tạo ra địa chỉ của ô nhớ cần truy cập	Do không phải mã hóa địa chỉ của bộ nhớ trong mã lệnh	Do dữ liệu sẽ được truyền vào thanh ghi địa chỉ	Cả ba câu kia đều đúng	3
109	Khi có hàng đợi lệnh (Instruction Queue) tốc độ thực hiện chương trình của CPU sẽ tăng lên do:		CPU có thể đồng thời thực hiện nhiều lệnh tại cùng một thời điểm	CPU có thể thực hiện hai tác vụ lấy lệnh và thực hiện lệnh tại cùng một thời điểm	CPU sẽ sắp hàng các lệnh của chương trình theo trình tự lập trình	Cả ba câu kia đều đúng	3
110	Khi truy cập một chuỗi dữ liệu trong bộ nhớ việc sử dụng các lệnh ngăn xếp sẽ có các lợi điểm gì?		Không cần thay đổi địa chỉ để truy cập tới các dữ liệu kế tiếp	Có thể truy cập nhiều dữ liệu trong chuỗi một cách đồng thời	Không cần truy cập dữ liệu một cách tuần tự	Không cần cung cấp địa chỉ bộ nhớ để truy cập dữ liệu	3
111	Khi truy cập một chuỗi dữ liệu trong bộ nhớ việc sử dụng các lệnh với chế độ chỉ số có các lợi điểm gì?		Không cần thay đổi địa chỉ để truy cập tới các dữ liệu kế tiếp	Có thể truy cập nhiều dữ liệu trong chuỗi một cách đồng thời	Có thể truy cập tới các phần tử khác nhau của chuỗi theo chỉ số trong lệnh, mà không cần theo địa chỉ cụ thể của nó	Không cần cung cấp địa chỉ bộ nhớ để truy cập dữ liệu	3

112	Tại sao cần giải mã lệnh?		Để với mã lệnh ngắn có thể cung cấp nhiều tín hiệu cho phép các khối mạch trong CPU thực hiện lệnh	Để bảo mật các mã lệnh trong chương trình	Để thực hiện các lệnh nhanh hơn	Cả ba câu kia đều đúng	3
113	Mã điều kiện PL xảy ra khi:		NF = 0	ZF = 1	NF = 1	VF = 1	3
114	Mã điều kiện HI xảy ra khi:		CF=1 và ZF=1	CF=1 và ZF=0	CF=0 và ZF=0	CF=0 và ZF=1	3
115	Mã điều kiện LS xảy ra khi:		CF=1 hoặc ZF=1	CF=1 hoặc ZF=0	CF=0 hoặc ZF=0	CF=0 hoặc ZF=1	3
116	Mã điều kiện GE xảy ra khi:		NF=VF	NF≠VF	ZF=VF	ZF≠VF	3
117	Mã điều kiện LT xảy ra khi:		NF=VF	NF≠VF	ZF=VF	ZF≠VF	3
118	Mã điều kiện LE xảy ra khi:		ZF=1 hoặc NF≠VF	ZF=1 và NF≠VF	CF=1 và NF≠VF	CF=1 và NF≠VF	3
119	Mã điều kiện LE xảy ra khi:		ZF=0 hoặc NF=VF	ZF=1 hoặc NF≠VF	ZF=1 hoặc NF=VF	ZF=0 hoặc NF≠VF	3
120	Mã điều kiện GT xảy ra khi:		ZF=1 và NF=VF	ZF=1 và NF≠VF	ZF=0 và NF=VF	ZF=0 và NF≠VF	3
121	Để điều khiển đọc một chip nhớ trong hệ thống vi xử lý cần:		Cấp tín hiệu chọn chip nhớ; Cấp địa chỉ để xác định ô nhớ cần đọc; Cấp tín hiệu cho phép đọc; Nhận dữ liệu ở data BUS	Cấp tín hiệu chọn chip nhớ; Chờ bộ nhớ cấp địa chỉ ô nhớ cần đọc; Cấp tín hiệu cho phép đọc; Nhận dữ liệu ở data BUS	Cấp tín hiệu chọn chip nhớ; Cấp địa chỉ để xác định ô nhớ cần đọc; Chờ tín hiệu cho phép đọc từ bộ nhớ; Nhận dữ liệu ở data BUS	Cấp tín hiệu chọn chip nhớ; Cấp địa chỉ để xác định ô nhớ cần đọc; Chờ tín hiệu cho phép đọc từ bộ nhớ;	1

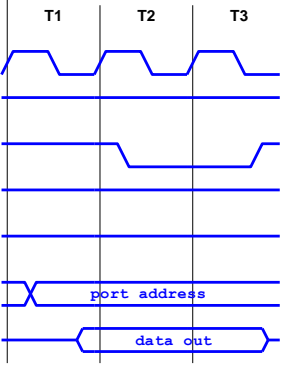
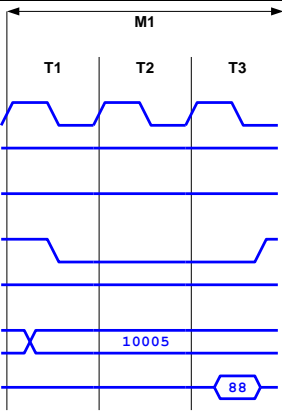
						Cấp dữ liệu ra data BUS	
122	Để điều khiển ghi một chip nhớ trong hệ thống vi xử lý cần:		Cấp tín hiệu chọn chip nhớ; Cấp địa chỉ để xác định ô nhớ cần ghi; Cấp dữ liệu vào data bus; Cấp tín hiệu điều khiển ghi	Cấp tín hiệu chọn chip nhớ; Cấp địa chỉ để xác định ô nhớ cần ghi; Cấp dữ liệu vào data bus; Chờ bộ nhớ cấp tín hiệu xác nhận đã ghi	Cấp tín hiệu chọn chip nhớ; Chờ bộ nhớ cấp địa chỉ ô nhớ cần ghi; Cấp dữ liệu vào data bus; Cấp tín hiệu điều khiển ghi	Cấp tín hiệu chọn chip nhớ; Cấp địa chỉ để xác định ô nhớ cần ghi; Cấp tín hiệu điều khiển ghi; Nhận dữ liệu từ data bus	1
123	Tại sao các hệ thống vi xử lý yêu cầu dung lượng bộ nhớ lớn thường sử dụng DRAM làm bộ nhớ chính?		Có giá thành rẻ	Tốc độ truy cập nhanh	Không cần chu kỳ làm tươi	Tiêu thụ ít năng lượng	1
124	Cache là loại bộ nhớ:		SRAM có tốc độ truy cập nhanh	DRAM có tốc độ truy cập nhanh	Bộ nhớ đệm giữa thiết bị ngoại vi và hệ thống vi xử lý	Flash có tốc độ truy cập nhanh	1
125	Các đặc tính khác nhau giữa DRAM và SRAM bao gồm:		Cấu tạo của tế bào nhớ	Tốc độ truy cập	Độ ổn định của dữ liệu	Cả ba câu kia đều đúng	1
126	Flash ROM là loại bộ nhớ:		Người sử dụng có thể ghi xóa dữ liệu nhiều lần bằng điện cả chip nhớ	Người sử dụng có thể ghi xóa dữ liệu nhiều lần bằng điện từng bit nhớ	Người sử dụng có thể ghi xóa dữ liệu nhiều lần bằng điện từng ô nhớ	Người sử dụng có thể ghi xóa dữ liệu nhiều lần bằng điện từng khối dữ liệu	1
127	EEPROM là loại bộ nhớ:		Dữ liệu được ghi vào trong khi sản xuất ra nó	Người sử dụng có thể ghi dữ liệu vào một lần	Người sử dụng có thể ghi xóa dữ liệu nhiều lần bằng điện từng ô nhớ	Người sử dụng có thể ghi dữ liệu vào bằng điện và xóa bằng tia cực tím	1

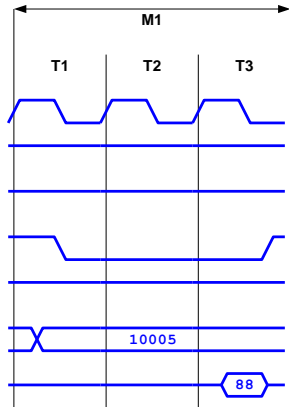
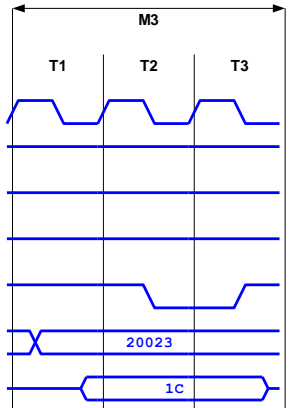
128	Loại bộ nhớ nào sau đây thường được sử dụng làm bộ nhớ ngoài?		HDD	Flash disk	SD card	Cả ba câu kia đều đúng	1
129	Trong phương thức điều khiển vào ra theo ngắt (Interrupt):		CPU dừng tiến trình hiện tại để chạy tiến trình phục vụ vào ra khi có yêu cầu	CPU dừng hoạt động sau khi phục vụ vào ra	Vào ra dừng CPU để tự thực hiện quá trình truyền dữ liệu với bộ nhớ	CPU tự động dừng hoạt động vào ra khi cần thiết	1
130	Trong phương thức điều khiển vào ra theo quét vòng (Polling) là:		CPU tuần tự thực hiện các lệnh để phục vụ vào ra	CPU tuần tự kiểm tra và thực hiện các tiến trình phục vụ các vào ra	CPU tuần tự truyền dữ liệu giữa các vào ra và bộ nhớ	CPU tuần tự chờ các vào ra hoạt động	1
131	Trong phương thức điều khiển vào ra theo truy cập bộ nhớ trực tiếp (DMA – Direct Memory Access):		CPU truyền dữ liệu với vào ra một cách trực tiếp	Vào ra trực tiếp điều khiển việc truyền dữ liệu với CPU	CPU dùng truy cập BUS để bộ nhớ và vào ra truyền dữ liệu trực tiếp với nhau	Bộ điều khiển DMA trực tiếp điều khiển hoạt động truyền dữ liệu với CPU	1
132	Trong các chế độ truyền dữ liệu nối tiếp đồng bộ, hai bên truyền nhận có thể đồng bộ với nhau bằng:		Thay đổi trạng thái đường truyền	Xung clock	Tín hiệu điều khiển đồng bộ cung cấp từ CPU	Tín hiệu điều khiển đồng bộ cung cấp từ bộ nhớ	1
133	Khi sử dụng cấu trúc Harvard tốc độ thực hiện các chương trình tăng lên do:		Tập lệnh đơn giản giảm thời gian giải mã lệnh	CPU có thể lấy lệnh và lấy dữ liệu tại cùng một thời điểm	Cấu trúc phân cứng đơn giản	Có cấu trúc đường ống cho phép thực hiện nhiều lệnh tại cùng một thời điểm	2
134			Ngõ ra cấp tín hiệu cho phép đọc dữ liệu chuyển tới CPU	Là tín hiệu hai chiều do CPU tác động khi đọc dữ liệu	Ngõ vào nhận tín hiệu đọc dữ liệu từ CPU	Là tín hiệu hai chiều do SRAM tác động khi đọc dữ liệu	2

	Trong định thời bộ nhớ SRAM ở trên, tín hiệu OE là:					
135	 <p>Trong định thời bộ nhớ ROM ở trên, khi CE ở mức cao:</p>	Bus địa chỉ và bus dữ liệu của ROM sẽ ở trạng thái trở kháng cao	Bus địa chỉ tích cực và bus dữ liệu của ROM sẽ ở trạng thái trở kháng cao	Bus dữ liệu tích cực và bus địa chỉ của ROM sẽ ở trạng thái trở kháng cao	Bus địa chỉ và bus dữ liệu của ROM sẽ ở trạng thái tích cực	2
136	 <p>Trong định thời (Timing diagram) ở trên, để thực hiện chu kỳ đọc dữ liệu cần:</p>	Cấp DT/R=1; cấp địa chỉ và chốt bằng tín hiệu ALE=1; cấp IO/M để xác định đọc bộ nhớ hay vào ra; cấp RD=0 tác động đọc bộ nhớ; cấp DEN=0 cho phép bộ đệm bus dữ liệu và lấy dữ liệu từ Data Bus vào CPU	Cấp DT/R=0; cấp địa chỉ và chốt bằng tín hiệu ALE=1; cấp IO/M để xác định đọc bộ nhớ hay vào ra; cấp RD=0 tác động đọc bộ nhớ; cấp DEN=0 cho phép bộ đệm bus dữ liệu và lấy dữ liệu từ Data Bus vào CPU	Cấp DT/R=0; cấp địa chỉ và chốt bằng tín hiệu ALE=0; cấp IO/M để xác định đọc bộ nhớ hay vào ra; cấp RD=0 tác động đọc bộ nhớ; cấp DEN=0 cho phép bộ đệm bus dữ liệu và lấy dữ liệu từ Data Bus vào CPU	Cấp DT/R=0; cấp địa chỉ và chốt bằng tín hiệu ALE=1; cấp IO/M để xác định đọc bộ nhớ hay vào ra; cấp RD=1 tác động đọc bộ nhớ; cấp DEN=0 cho phép bộ đệm bus dữ liệu và lấy dữ liệu từ Data Bus vào CPU	2

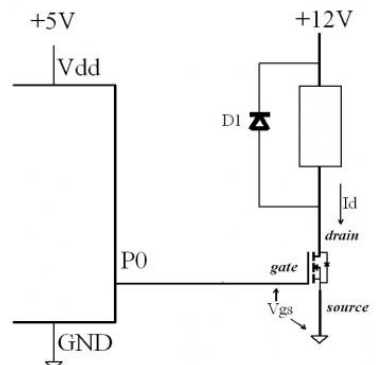
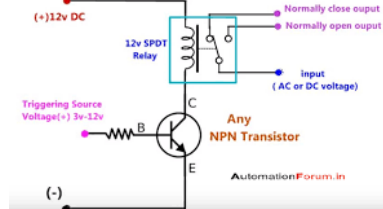
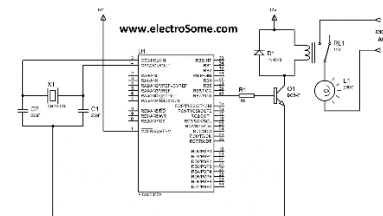
137	 <p>Trong định thời (Timing diagram) ở trên, để thực hiện chu kỳ ghi dữ liệu cần:</p>	Cấp DT/R=1; cấp địa chỉ và chốt bằng tín hiệu ALE=1; cấp IO/M để xác định đọc bộ nhớ hay vào ra; cấp RD=0 tác động đọc bộ nhớ; cấp DEN=0 cho phép bộ đệm bus dữ liệu và lấy dữ liệu từ Data Bus vào CPU	Cấp DT/R=0; cấp địa chỉ và chốt bằng tín hiệu ALE=1; cấp IO/M để xác định đọc bộ nhớ hay vào ra; cấp RD=0 tác động đọc bộ nhớ; cấp DEN=0 cho phép bộ đệm bus dữ liệu và lấy dữ liệu từ Data Bus vào CPU	Cấp DT/R=0; cấp địa chỉ và chốt bằng tín hiệu ALE=0; cấp IO/M để xác định đọc bộ nhớ hay vào ra; cấp RD=0 tác động đọc bộ nhớ; cấp DEN=0 cho phép bộ đệm bus dữ liệu và lấy dữ liệu từ Data Bus vào CPU	Cấp DT/R=0; cấp địa chỉ và chốt bằng tín hiệu ALE=1; cấp IO/M để xác định đọc bộ nhớ hay vào ra; cấp RD=1 tác động đọc bộ nhớ; cấp DEN=0 cho phép bộ đệm bus dữ liệu và lấy dữ liệu từ Data Bus vào CPU	2
138	 <p>Trong sơ đồ kết nối hệ thống vi xử lý 8086 trên tín hiệu ALE và các bộ chốt (Latch) được sử dụng để:</p>	Chốt BUS điều khiển kết nối tới bộ nhớ và vào ra	Chốt BUS địa chỉ kết nối tới bộ nhớ và vào ra	Chốt BUS dữ liệu kết nối tới bộ nhớ và vào ra	Điều khiển cấp nguồn cho hệ thống vi xử lý	2
139		Khi CPU đọc bộ nhớ	Khi CPU truy cập bộ nhớ	Khi CPU ghi bộ nhớ	Khi CPU truy cập vào ra	2

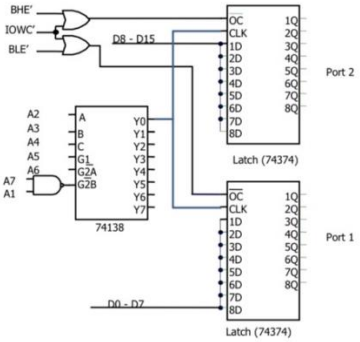
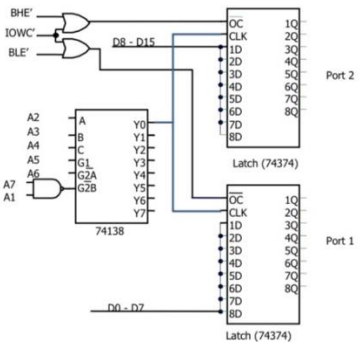
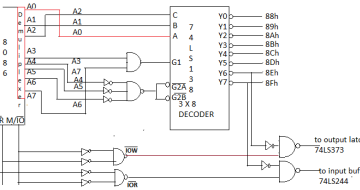
	Trong sơ đồ kết nối hệ thống vi xử lý hình vẽ trên, tín hiệu IO/M sẽ có mức logic 0:																																																													
140	<table><tr><th>WE</th><th>CS1</th><th>CS2</th><th>OE</th><th>Mode</th><th>V_{cc} current</th><th>IO pin</th><th>Ref. cycle</th></tr><tr><td>x</td><td>H</td><td>x</td><td>x</td><td>Not selected (power down)</td><td>I_{DD1}, I_{DD2}</td><td>High-Z</td><td>---</td></tr><tr><td>x</td><td>x</td><td>L</td><td>x</td><td>Not selected (power down)</td><td>I_{DD1}, I_{DD2}</td><td>High-Z</td><td>---</td></tr><tr><td>H</td><td>L</td><td>H</td><td>H</td><td>Output disable</td><td>I_{DD1}</td><td>High-Z</td><td>---</td></tr><tr><td>H</td><td>L</td><td>H</td><td>L</td><td>Read</td><td>I_{DD1}</td><td>Dout</td><td>Read cycle (1)-(3)</td></tr><tr><td>L</td><td>L</td><td>H</td><td>H</td><td>Write</td><td>I_{DD1}</td><td>Din</td><td>Write cycle (1)</td></tr><tr><td>L</td><td>L</td><td>H</td><td>L</td><td>Write</td><td>I_{DD1}</td><td>Din</td><td>Write cycle (2)</td></tr></table> <p>Note: x: H or L</p> <p>Theo bảng trạng thái của bộ nhớ SRAM ở trên, dữ liệu sẽ được lưu vào SRAM khi:</p>	WE	CS1	CS2	OE	Mode	V _{cc} current	IO pin	Ref. cycle	x	H	x	x	Not selected (power down)	I _{DD1} , I _{DD2}	High-Z	---	x	x	L	x	Not selected (power down)	I _{DD1} , I _{DD2}	High-Z	---	H	L	H	H	Output disable	I _{DD1}	High-Z	---	H	L	H	L	Read	I _{DD1}	Dout	Read cycle (1)-(3)	L	L	H	H	Write	I _{DD1}	Din	Write cycle (1)	L	L	H	L	Write	I _{DD1}	Din	Write cycle (2)	CS1=0; CS2=1;WE=0; OE=0	CS1=0; CS2=1;WE=1; OE=1	CS1=0; CS2=1;WE=1; OE=0	Cả ba câu kia đều đúng	2
WE	CS1	CS2	OE	Mode	V _{cc} current	IO pin	Ref. cycle																																																							
x	H	x	x	Not selected (power down)	I _{DD1} , I _{DD2}	High-Z	---																																																							
x	x	L	x	Not selected (power down)	I _{DD1} , I _{DD2}	High-Z	---																																																							
H	L	H	H	Output disable	I _{DD1}	High-Z	---																																																							
H	L	H	L	Read	I _{DD1}	Dout	Read cycle (1)-(3)																																																							
L	L	H	H	Write	I _{DD1}	Din	Write cycle (1)																																																							
L	L	H	L	Write	I _{DD1}	Din	Write cycle (2)																																																							
141	<div><p>Trong định thời đọc công vào ra hình trên, trong chu kỳ đọc vào ra các tín hiệu MEMR và MEMW sẽ ở trạng thái:</p></div>	Logic 0	Logic 1	High Z	MEMR mức 0 MEMW mức 1	2																																																								

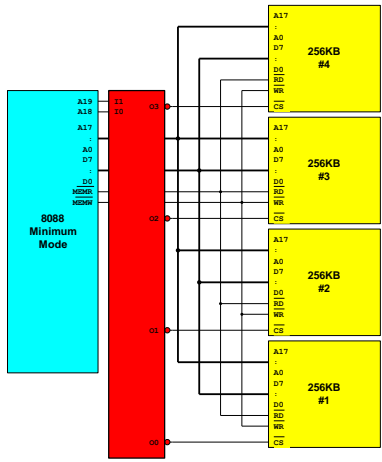
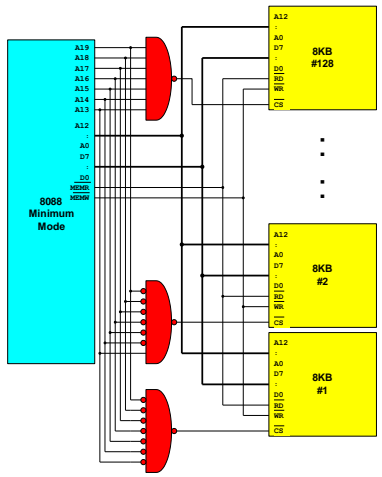
142	<div><p>Trong định thời đọc công vào ra hình trên, trong chu kỳ ghi vào ra các tín hiệu MEMR và MEMW sẽ ở trạng thái:</p></div>	Logic 0	Logic 1	High Z	MEMR mức 0 MEMW mức 1	2
143	<div><p>Giản đồ thời gian hình vẽ trên là giản đồ của chu kỳ:</p></div>	Đọc bộ nhớ	Ghi bộ nhớ	Đọc vào ra	Ghi vào ra	2

144	<div><p>Trong định thời hình trên, trong chu kỳ ghi vào ra các tín hiệu IOR và IOW sẽ ở trạng thái:</p></div>	Logic 0	Logic 1	High Z	MEMR mức 0 MEMW mức 1	2																																																								
145	<div><p>Trong định thời hình trên, trong chu kỳ ghi vào ra các tín hiệu IOR và IOW sẽ ở trạng thái:</p></div>	Logic 0	Logic 1	High Z	MEMR mức 0 MEMW mức 1	2																																																								
146	<table><tr><th>Mode</th><th>\bar{E}</th><th>\bar{G}</th><th>\bar{P}</th><th>A0</th><th>V_{pp}</th><th>Q0 - Q7</th></tr><tr><td>Read</td><td>V_L</td><td>V_L</td><td>V_H</td><td>X</td><td>V_{CC}</td><td>Data Out</td></tr><tr><td>Output Disable</td><td>V_L</td><td>V_H</td><td>V_H</td><td>X</td><td>V_{CC}</td><td>Hi-Z</td></tr><tr><td>Program</td><td>V_L</td><td>V_H</td><td>V_L Pulse</td><td>X</td><td>V_{pp}</td><td>Data In</td></tr><tr><td>Verify</td><td>V_L</td><td>V_L</td><td>V_H</td><td>X</td><td>V_{pp}</td><td>Data Out</td></tr><tr><td>Program Inhibit</td><td>V_H</td><td>X</td><td>X</td><td>X</td><td>V_{pp}</td><td>Hi-Z</td></tr><tr><td>Standby</td><td>V_H</td><td>X</td><td>X</td><td>X</td><td>V_{CC}</td><td>Hi-Z</td></tr><tr><td>Electronic Signature</td><td>V_L</td><td>V_L</td><td>V_H</td><td>V_{IO}</td><td>V_{CC}</td><td>Codes Out</td></tr></table> <p>Note: X = V_{IO} or V_L, V_{IO} = 12V ± 0.5%.</p>	Mode	\bar{E}	\bar{G}	\bar{P}	A0	V _{pp}	Q0 - Q7	Read	V _L	V _L	V _H	X	V _{CC}	Data Out	Output Disable	V _L	V _H	V _H	X	V _{CC}	Hi-Z	Program	V _L	V _H	V _L Pulse	X	V _{pp}	Data In	Verify	V _L	V _L	V _H	X	V _{pp}	Data Out	Program Inhibit	V _H	X	X	X	V _{pp}	Hi-Z	Standby	V _H	X	X	X	V _{CC}	Hi-Z	Electronic Signature	V _L	V _L	V _H	V _{IO}	V _{CC}	Codes Out	Ngõ vào E	Ngõ vào P	Ngõ vào G	Ngõ vào V _{pp}	2
Mode	\bar{E}	\bar{G}	\bar{P}	A0	V _{pp}	Q0 - Q7																																																								
Read	V _L	V _L	V _H	X	V _{CC}	Data Out																																																								
Output Disable	V _L	V _H	V _H	X	V _{CC}	Hi-Z																																																								
Program	V _L	V _H	V _L Pulse	X	V _{pp}	Data In																																																								
Verify	V _L	V _L	V _H	X	V _{pp}	Data Out																																																								
Program Inhibit	V _H	X	X	X	V _{pp}	Hi-Z																																																								
Standby	V _H	X	X	X	V _{CC}	Hi-Z																																																								
Electronic Signature	V _L	V _L	V _H	V _{IO}	V _{CC}	Codes Out																																																								

	Trong định thời ở trên, dữ liệu được ghi vào bộ nhớ EPROM khi cấp xung lập trình vào:																																																													
147	<table border="1"><thead><tr><th>Mode</th><th>\bar{E}</th><th>\bar{G}</th><th>\bar{P}</th><th>AB</th><th>Vpp</th><th>Q0-Q7</th></tr></thead><tbody><tr><td>Read</td><td>V_L</td><td>V_L</td><td>V_{Hi}</td><td>X</td><td>V_{CC}</td><td>Data Out</td></tr><tr><td>Output Disable</td><td>V_L</td><td>V_{Hi}</td><td>V_{Hi}</td><td>X</td><td>V_{CC}</td><td>Hi-Z</td></tr><tr><td>Program</td><td>V_L</td><td>V_{Hi}</td><td>V_L Pulse</td><td>X</td><td>V_{pp}</td><td>Data In</td></tr><tr><td>Verify</td><td>V_L</td><td>V_L</td><td>V_{Hi}</td><td>X</td><td>V_{pp}</td><td>Data Out</td></tr><tr><td>Program Inhibit</td><td>V_{Hi}</td><td>X</td><td>X</td><td>X</td><td>V_{pp}</td><td>Hi-Z</td></tr><tr><td>Standby</td><td>V_{Hi}</td><td>X</td><td>X</td><td>X</td><td>V_{CC}</td><td>Hi-Z</td></tr><tr><td>Electronic Signature</td><td>V_L</td><td>V_L</td><td>V_{Hi}</td><td>V_{CC}</td><td>V_{CC}</td><td>Codes Out</td></tr></tbody></table> <p>Note: X = 1μs or V_L, V_{Hi} = 12V ± 0.5%</p> <p>Trong định thời ở trên, tín hiệu G là:</p>	Mode	\bar{E}	\bar{G}	\bar{P}	AB	Vpp	Q0-Q7	Read	V _L	V _L	V _{Hi}	X	V _{CC}	Data Out	Output Disable	V _L	V _{Hi}	V _{Hi}	X	V _{CC}	Hi-Z	Program	V _L	V _{Hi}	V _L Pulse	X	V _{pp}	Data In	Verify	V _L	V _L	V _{Hi}	X	V _{pp}	Data Out	Program Inhibit	V _{Hi}	X	X	X	V _{pp}	Hi-Z	Standby	V _{Hi}	X	X	X	V _{CC}	Hi-Z	Electronic Signature	V _L	V _L	V _{Hi}	V _{CC}	V _{CC}	Codes Out	Ngõ vào cung cấp tín hiệu đọc dữ liệu từ EPROM	Ngõ vào cung cấp tín hiệu ghi dữ liệu vào EPROM	Ngõ vào cung cấp tín hiệu cho phép EPROM hoạt động	Ngõ vào cung cấp tín hiệu cấm EPROM hoạt động	2
Mode	\bar{E}	\bar{G}	\bar{P}	AB	Vpp	Q0-Q7																																																								
Read	V _L	V _L	V _{Hi}	X	V _{CC}	Data Out																																																								
Output Disable	V _L	V _{Hi}	V _{Hi}	X	V _{CC}	Hi-Z																																																								
Program	V _L	V _{Hi}	V _L Pulse	X	V _{pp}	Data In																																																								
Verify	V _L	V _L	V _{Hi}	X	V _{pp}	Data Out																																																								
Program Inhibit	V _{Hi}	X	X	X	V _{pp}	Hi-Z																																																								
Standby	V _{Hi}	X	X	X	V _{CC}	Hi-Z																																																								
Electronic Signature	V _L	V _L	V _{Hi}	V _{CC}	V _{CC}	Codes Out																																																								
148	<p>Trong sơ đồ kết nối IO hình vẽ trên, vi mạch 16L8 thực hiện chức năng:</p>	Giải mã địa chỉ	Chốt địa chỉ	Truyền dữ liệu	Cung cấp tín hiệu điều khiển vào ra	2																																																								
149	<p>Thiết bị điện tử hình trên có chức năng:</p>	Cách ly quang giữa ngõ ra và tải	Cho phép điều khiển tải xoay chiều	Tăng công suất điều khiển cho ngõ ra	Cả ba câu kia đều đúng	2																																																								

150	 <p>Trong sơ đồ trên, MOSFET có chức năng:</p>	Cấp dòng điện đủ lớn cho tải	Thay đổi mức logic điều khiển tải	Cho phép điều khiển tải có dòng điện lớn bằng tín hiệu điều khiển có dòng điện nhỏ cung cấp từ ngõ ra	Cả ba câu kia đều đúng	2
151	 <p>Mạch giao tiếp relay hình vẽ trên có chức năng:</p>	Cho phép điều khiển tải xoay chiều	Cho phép điều khiển tải một chiều	Cách ly giữa mạch công suất và mạch điều khiển	Cả ba câu kia đều đúng	2
152	 <p>Trong sơ đồ hình vẽ trên diode D1 có chức năng:</p>	Xả dòng trên cuộn dây relay khi transistor ngưng dẫn tránh hư hỏng transistor	Dẫn dòng điện khi ngõ ra điều khiển cung cấp mức logic 1	Tăng dòng điện cung cấp cho tải	Cả ba câu kia đều đúng	2

153	 <p>Figure 2</p> <p>Trong sơ đồ trên, PORT1 sẽ có địa chỉ:</p>	A0H	A1H	A4	A2	3
154	 <p>Figure 2</p> <p>Trong sơ đồ trên, PORT1 sẽ có địa chỉ:</p>	A0H	A1H	A4H	A3H	3
155	 <p>Trong sơ đồ trên để truy cập tới các cổng vào ra cho phép từ bộ giải mã địa</p>	00B	01B	10B	11B	3

	Trong sơ đồ kết nối bộ nhớ hình vẽ trên bộ nhớ #2 có vùng địa chỉ:						
159	 <p>Trong sơ đồ kết nối bộ nhớ hình vẽ trên bộ nhớ #3 có vùng địa chỉ:</p>		00000H-3FFFFH	C0000H-FFFFFH	80000H-BFFFFH	40000H-7FFFFH	3
160	 <p>Trong sơ đồ kết nối bộ nhớ hình vẽ trên sẽ có bao nhiêu bộ nhớ 8KB?</p>		56	128	256	512	3

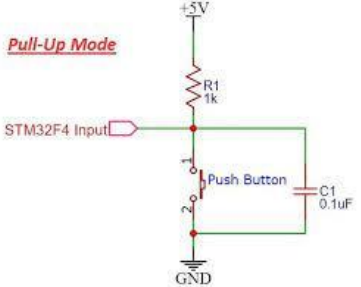
161	GPIO (General Purpose Input/Output) là khối chức năng gì trong vi điều khiển?		Giao tiếp vào ra số	Biến đổi ADC	Xử lý ngắt	Đa chức năng	1
162	Trong thư viện HAL (Hardware Abstraction Layer) cho STM32 HAL_Init() được sử dụng để:		Khởi động các phần cứng	Cấu hình xung nhịp	Khởi động cấu hình các cấu trúc dữ liệu bên trong thư viện	Cả ba câu kia đều đúng	1
163	Trong thư viện HAL (Hardware Abstraction Layer) cho STM32 GPIO_InitTypeDef là:		Cấu trúc dữ liệu để khởi động GPIO	Hàm để khởi động GPIO	Truyền thông tin để khởi động GPIO	Cả ba câu kia đều đúng	1
164	Trong thư viện HAL cho STM32 GPIO_InitStruct.Pin, GPIO_InitStruct.Mode, GPIO_InitStruct.Pull, GPIO_InitStruct.Speed là:		Cấu trúc dữ liệu để khởi động GPIO	Hàm để khởi động GPIO	Truyền thông tin để khởi động GPIO	Cả ba câu kia đều đúng	1
165	Trong thư viện HAL cho STM32 giá trị GPIO_Mode_IN để:		Nhận giá trị từ chân GPIO	Cấu hình GPIO làm chân vào	Thay đổi giá trị ngõ vào của GPIO	Tạo chế độ vào cho tất cả GPIO	1
166	Trong thư viện HAL cho STM32 giá trị GPIO_Mode_OUT để:		Cấp giá trị cho chân GPIO	Cấu hình GPIO làm chân ra	Thay đổi giá trị ngõ ra của GPIO	Tạo chế độ ra cho tất cả GPIO	1
167	Trong thư viện HAL cho STM32 giá trị GPIO_Mode_AF để:		Thay đổi giá trị cung cấp cho chân GPIO	Cấu hình GPIO thay đổi chức năng giữa vào và ra	Cấu hình cho phép chân GPIO sử dụng cho các chức năng khác như USART, SPI, I2C, ...	Cấu hình cho chân GPIO là đường vào ra tương tự	1

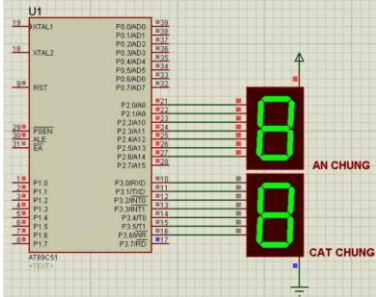
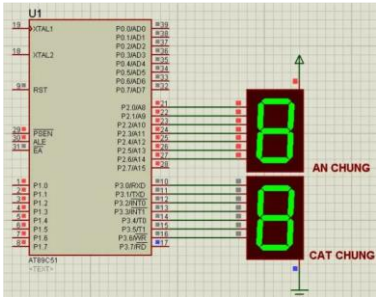
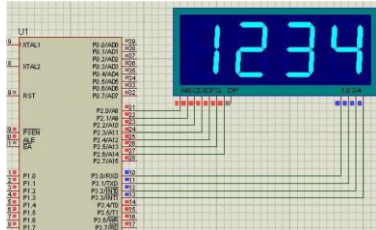
168	Trong thư viện HAL cho STM32 giá trị GPIO_Mode_ANALOG để:		Cung cấp tín hiệu tương tự ra chân GPIO	Nhận tín hiệu tương tự từ chân GPIO	Cấu hình cho phép chân GPIO làm chân nhận hoặc cấp tín hiệu tương tự	Cả ba câu kia đều đúng	1
169	Trong thư viện HAL cho STM32 giá trị GPIO_Mode_IT để:		Cấu hình cho phép chân GPIO làm chân vào trở kháng cao	Cấu hình cho phép chân GPIO làm chân truyền thông tin	Cấu hình cho phép chân GPIO làm chân nhận thông tin	Cấu hình cho phép chân GPIO làm chân nhận tín hiệu yêu cầu ngắt	1
170	Trong thư viện HAL cho STM32 biến GPIO_InitStruct.Pull để:		Xác định cách kết nối chân GPIO với điện trở kéo lên hoặc kéo xuống	Xác định cách kết nối chân GPIO với điện trở kéo lên	Xác định cách kết nối chân GPIO với điện trở kéo xuống	Xác định cách kết nối chân GPIO không có điện trở kéo lên hoặc kéo xuống	1
171	Trong thư viện HAL cho STM32 biến GPIO_InitStruct.Pin để:		Xác định chân GPIO sẽ được cấu hình	Xác định chân GPIO là ngõ ra	Xác định chân GPIO là ngõ vào	Xác định chân GPIO là trở kháng cao	1
172	Trong thư viện HAL cho STM32 hàm HAL_GPIO_Init() cần truyền biến gì:		Biến xác định chân cần khởi động; Biến xác định cấu hình chân cần khởi động	Biến xác định cổng cần khởi động; Biến xác định cấu hình các chân cần khởi động	Biến xác định các chân cần khởi động; Biến xác định cấu hình các chân cần khởi động	Cả ba câu kia đều sai	1
173	Các bit CNF trong thanh ghi CRL của STM32 sử dụng để:		Lập xóa bit các chân GPIO	Cấu hình tốc độ của các chân GPIO	Cấu hình chế độ vào ra của các chân GPIO	Cấu hình điện trở kéo lên kéo xuống của các chân GPIO	1
174	Các bit MODE trong thanh ghi CRL của STM32 sử dụng để:		Lập xóa bit các chân GPIO	Cấu hình tốc độ của các chân GPIO	Cấu hình chế độ vào ra của các chân GPIO	Cấu hình điện trở kéo lên kéo	1

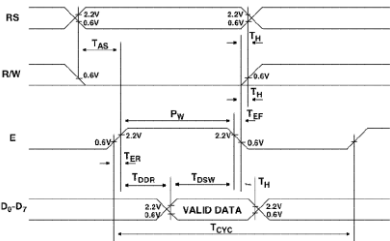
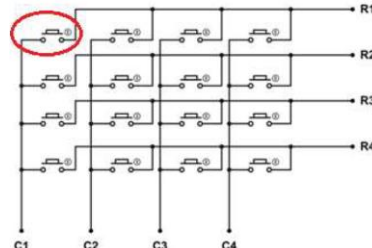
						xuống của các chân GPIO	
175	Thanh ghi GPIOB->ORD của STM32:		Là thanh ghi sử dụng 16 bit thấp cung cấp dữ liệu ra PORTB	Là thanh ghi sử dụng 16 bit cao cung cấp dữ liệu ra PORTB	Là thanh ghi sử dụng 32 cung cấp dữ liệu ra PORTB	Là thanh ghi sử dụng 32 nhận dữ liệu PORTB	1
176	Thanh ghi GPIOA->BSRR của STM32:		Dùng 16 bit thấp để reset các bit Port A về 0; 16 bit cao set các bit Port A lên 1	Dùng 16 bit cao để reset các bit Port A về 0; 16 bit thấp set các bit Port A lên 1	Là thanh ghi để reset các bit Port A về 0	Là thanh ghi để set các bit Port A lên 1	1
177	Thanh ghi GPIOB->IDR của STM32:		Sử dụng 32 bit cao để chứa dữ liệu nhận được từ các chân vào PORTB	Sử dụng 32 bit thấp để chứa dữ liệu nhận được từ các chân vào PORTB	Sử dụng 16 bit cao để chứa dữ liệu nhận được từ các chân vào PORTB	Sử dụng 16 bit thấp để chứa dữ liệu nhận được từ các chân vào PORTB	1
178	HAL_GPIO_WritePin, (GPIOx,GPIO_Pin,PinState) Tham số GPIO_Pin PinState trong hàm HAL STM32 trên:		Có các giá trị GPIO_PIN_0 và GPIO_PIN_1	Có các giá trị GPIO_PIN_HI và GPIO_PIN_LO	Có các giá trị GPIO_PIN_RESET và GPIO_PIN_SET	Cả ba câu kia đều đúng	1
179	Trong hàm HAL_GPIO_ReadPin() thư viện HAL cho STM32 sẽ có các tham số chỉ thị:		Cổng và cấu hình của chân GPIO	Cổng và cấu hình của cổng	Cổng và chân GPIO của cổng	Cả ba câu kia đều đúng	1
180	Tín hiệu RS của text LCD sử dụng để:		Chọn thanh ghi dữ liệu hoặc điều khiển khi ghi dữ liệu vào LCD	Chọn thanh ghi dữ liệu hoặc điều khiển khi đọc dữ liệu từ LCD	Chọn thanh ghi trạng thái hoặc điều khiển khi ghi dữ liệu vào LCD	Chọn thanh ghi trạng thái hoặc điều khiển khi đọc dữ liệu từ LCD	1

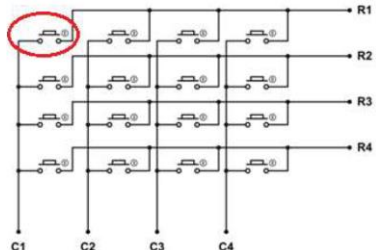
181	Trước khi gửi dữ liệu hiển thị lên LCD có thể khởi động bằng các lệnh nào?		0x38; 0x0E	0x06; 0xC0	0x10; 0x14	0x18; 0x1C	1
182	Các thanh ghi cho ADC của STM32 là:		ADC_CR1,2 (Control Register 1,2) ; ADC_SMPR (Sample Time Register)	ADC_DR (Data Register); ADC_CCR (Common Control Register)	ADC_ISR (Status Register) và ADC_IER (Interrupt Enable Register)	Cả ba câu kia đều đúng	1
183	Các thanh ghi sử dụng cho I2C của STM32 bao gồm:		I2C_CR1; I2C_CR2; I2C_OAR1; I2C_OAR2; I2C_DR; I2C_SR1; I2C_SR2; I2C_CCR; I2C_TRISE; I2C_FLTR	I2C_CR1; I2C_CR2; I2C_OAR; I2C_DR1; I2C_DR2; I2C_SR1; I2C_SR2; I2C_CCR; I2C_TRISE; I2C_FLTR	I2C_CR1; I2C_CR2; I2C_OAR1; I2C_OAR2; I2C_OAR3; I2C_DR; I2C_SR1; I2C_SR2; I2C_CCR1; I2C_CCR2; I2C_TRISE; I2C_FLTR	I2C_CR1; I2C_OAR1; I2C_OAR2; I2C_OAR3; I2C_DR; I2C_SR1; I2C_SR2; I2C_CCR; I2C_TRISE; I2C_FLTR	1
184	Các thanh ghi sử dụng cho USART của STM32 bao gồm:		USART_CR1; USART_CR2; USART_BRR; USART_DR; USART_SR1; USART_SR2	USART_CR1; USART_CR2; USART_BRR1; USART_BRR2; USART_DR; USART_SR	USART_CR1; USART_CR2; USART_CR3; USART_BRR; USART_DR; USART_SR	USART_CR1; USART_CR2; USART_BRR; USART_DR1; USART_DR2; USART_SR	1
185	Các tham số khởi động (Initialization) cho một GPIO		Khởi động là ngõ vào hoặc ngõ ra, giá trị ban đầu	Khởi động Pull Up, Pull Down, Open, Floating, ...	Khởi động các chức năng: Ngắt, vào ra ADC, tạo xung, ...	Cả ba câu kia đều đúng	2

186	Các chức năng thường thấy của LED trong các hệ thống nhúng:		Thông báo hệ thống đã được cấp nguồn	Thông báo trạng hoạt động: mở, đóng, ...	Thông báo chế độ hoạt động: chạy, dừng, lỗi, ...	Cả ba câu kia đều đúng	2
187	Các chức năng thường thấy của nút nhấn trong các hệ thống nhúng:		Thực hiện tác vụ điều khiển, xác nhận hành động	Nhập tham số, dữ liệu hoạt động	Thiết lập cấu hình: chế độ, tần số, ...	Cả ba câu kia đều đúng	2
188	Các chức năng thường thấy của LED 7 đoạn trong các hệ thống nhúng:		Hiển thị các con số cho: đồng hồ, dữ liệu đo lường, kết quả phép tính, giá trị tham số hệ thống, số lượng sản phẩm ...	Hiển thị ký tự thông báo: trạng thái hệ thống, chế độ hoạt động, ...	Hiển thị các bản báo cho: Tên sản phẩm, thông báo, ...	Cả ba câu kia đều đúng	2
189	So sánh ưu nhược điểm của LCD so với LED 7 đoạn trong các hệ thống nhúng:		Hiển thị văn bản đồ họa ít phong phú, ít màu sắc hơn	Tiêu thụ công cao hơn	Đáp ứng nhanh hơn	Góc nhìn rộng hơn	2
190	GPIOB->CRL=(3<<0) (3<<4) (3<<8) (3<<12) Dòng lệnh lập trình thanh ghi CRL của STM32 ở trên thực hiện:		Cấu hình các chân PB0; PB1; PB2; PB3 của STM32 ở chế độ output push pull max speed 50Hz	Cấu hình các chân PB0; PB4; PB8; PB12 của STM32 ở chế độ output push pull max speed 50Hz	Cấu hình các chân PB0; PB1; PB2; PB3 của STM32 ở chế độ output open drain max speed 50Hz	Cấu hình các chân PB0; PB4; PB8; PB12 của STM32 ở chế độ output open drain max speed 50Hz	2
191	GPIOA->CRL=(8<<0) (8<<4) (8<<8) Dòng lệnh lập trình thanh ghi CRL của STM32 ở trên thực hiện:		Cấu hình chân PA0,PA1,PA2 input pull down	Cấu hình chân PA0,PA1,PA2 input pull up	Cấu hình chân PA0,PA1,PA2 input floating	Cả ba câu kia đều đúng	2

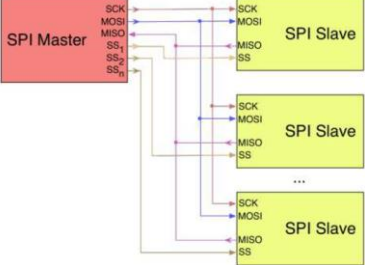
192	<p>$\text{GPIOC} \rightarrow \text{ODR} = (1 \ll 7) (1 \ll 8) (1 \ll 9)$</p> <p>Dòng lệnh lập trình thanh ghi ODR của STM32 ở trên thực hiện:</p>		Cung cấp bit 0 ra các chân 7, 8, 9 của PORTC	Cung cấp bit 1 ra các chân 7, 8, 9 của PORTC	Cấu hình các chân 7, 8, 9 PORTC là ngõ ra	Cấu hình các chân 7, 8, 9 PORTC là ngõ vào	2
193	<p>$\text{GPIOC} \rightarrow \text{BSRR} = (1 \ll 23) (8 \ll 24) (8 \ll 25)$</p> <p>Dòng lệnh lập trình thanh ghi BSRR của STM32 ở trên thực hiện:</p>		Đề đưa chân 7, 8, 9 PORTC lên mức cao	Đề đưa chân 23, 24, 25 PORTC lên mức cao	Đề đưa chân 23, 24, 25 PORTC về mức thấp	Đề đưa chân 7, 8, 9 PORTC về mức thấp	2
194	<p>Một LED gắn vào cổng ra vi điều khiển nối tiếp với 1 điện trở, biết điện áp mức cao ngõ ra là 5V, điện áp rơi trên LED khi sáng là 3V, dòng qua LED là 20mA, điện trở cần có giá trị bằng bao nhiêu?</p>		100Ω	150Ω	200Ω	250Ω	2
195	 <p>Trong hình kết nối nút nhấn vào STM32 hình vẽ trên, điện trở kéo lên (Pull up) có chức năng gì?</p>		Cấp nguồn cho nút nhấn hoạt động	Để chia dòng với điện trở kéo lên bên trong STM32	Cung cấp mức logic 1 cho STM32 khi nút không nhấn	Để nạp điện cho tụ	2

196	 <p>Trong sơ đồ kết nối LED 7 đoạn với vi điều khiển ở trên, muốn sáng số 5 trên LED 7 đoạn Anode chung cần cấp ra cổng điều khiển dữ liệu bao nhiêu?</p>	01101001B	10010110B	11001001B	01100110B	2
197	 <p>Trong sơ đồ kết nối LED 7 đoạn với vi điều khiển ở trên, muốn sáng số 5 trên LED 7 đoạn Kathode chung cần cấp ra cổng điều khiển dữ liệu bao nhiêu?</p>	01101001B	10010110B	11001001B	01100110B	2
198		1	2	3	4	2

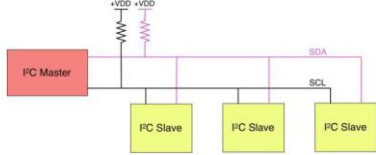
	<p>Trong sơ đồ kết nối LED 7 đoạn với vi điều khiển ở trên, tại một thời điểm vi điều khiển cung cấp dữ liệu cho mấy LED 7 đoạn?</p>					
199	 <p>Trong giản đồ thời gian của Text LCD hình trên, dữ liệu sẽ được ghi vào LCD ở:</p>	Cạnh lên của tín hiệu E	Cạnh xuống của tín hiệu E	Cạnh xuống của tín hiệu RS	Cạnh xuống của tín hiệu RW	2
200	 <p>Trong một bàn phím ma trận hình trên. Nếu kéo điện trở Pull Up ở tất cả các cột và cấp mức 0 ra các cột. Nếu không có phím nhấn dữ liệu đọc từ các hàng sẽ bằng:</p>	$R_4R_3R_2R_1 = 0000B$	$R_4R_3R_2R_1 = 1111B$	$R_4R_3R_2R_1 = 0001B$	$R_4R_3R_2R_1 = 1000B$	2

201	 <p>Trong một bàn phím ma trận hình trên. Nếu kéo điện trở Pull Up ở tất cả các cột và cấp mức 0 ra các cột. Nếu nhấn phím trên cùng bên trái (khoanh tròn) dữ liệu đọc từ các hàng sẽ bằng:</p>		$R_4R_3R_2R_1 = 0000B$	$R_4R_3R_2R_1 = 1111B$	$R_4R_3R_2R_1 = 0001B$	$R_4R_3R_2R_1 = 1000B$	2
202	Các tham số thường khởi động cho ADC là:		Tự động chuyển kênh (Auto Injected Mode); Ngưỡng chuyển đổi ngõ vào (Trigger Threshold); Nguồn đầu vào (Input Source)	Chế độ DMA (DMA Mode); Chế độ ngắt (Interrupt Mode); Chế độ quét (Scan Mode)	Tần số lấy mẫu (Sampling Frequency); Độ phân giải (Resolution); Định dạng dữ liệu đầu ra (Data Format)	Cả ba câu kia đều đúng	2
203	Trong thư viện HAL của STM32 hàm HAL_ADC_Init() sử dụng khởi động các tham số:		Chọn khối ADC; Chọn Clock; Chọn độ phân giải	Cân chỉnh nơi chứa dữ liệu trên thanh ghi dữ liệu; Chọn chế độ quét	Chọn chế độ biến đổi; Chọn số lượng kênh ADC sử dụng	Cả ba câu kia đều đúng	2
204	Để sử dụng ADC của STM32 theo chế độ Polling có thể sử dụng các hàm HAL theo trình tự:		HAL_ADC_Init(); HAL_ADC_ConfigChannel; HAL_ADC_Start;	HAL_ADC_Start; HAL_ADC_Init(); HAL_ADC_ConfigChannel;	HAL_ADC_Init(); HAL_ADC_ConfigChannel; HAL_ADC_Start;	HAL_ADC_Init(); HAL_ADC_ConfigChannel;	2

			HAL_ADC_GetValue; HAL_ADC_PollForConversion; HAL_ADC_Stop	HAL_ADC_PollForConversion; HAL_ADC_GetValue; HAL_ADC_Stop	HAL_ADC_PollForConversion; HAL_ADC_GetValue; HAL_ADC_Stop	HAL_ADC_PollForConversion; HAL_ADC_Start; HAL_ADC_GetValue; HAL_ADC_Stop	
205	Các thanh ghi thường sử dụng của khối SPI của STM32 bao gồm:		SPI_CR1; SPI_CR2; SPI_SS; SPI_DR	SPI_CT1; SPI_CT2; SPI_SR; SPI_DR	SPI_CR1; SPI_CR2; SPI_SR; SPI_DR	SPI_CR1; SPI_CR2; SPI_ST; SPI_DR	2
206	Các thông số khởi động cho cổng SPI của STM32 bao gồm:		Chọn cổng SPI; Chọn chế độ Master/Slaver; Số dây truyền; Kích thước dữ liệu truyền; Pha và cực của clock; Tần số Clock; Chọn số Slaver truyền dữ liệu với Master; Truyền MSB hay LSB trước	Chọn số lượng cổng SPI; Chọn chế độ Master/Slaver; Số dây truyền; Kích thước dữ liệu truyền; Pha và cực của clock; Tần số Clock; Chọn Slaver bằng phần cứng hay phần mềm; Truyền MSB hay LSB trước	Chọn cổng SPI; Chọn chế độ Master/Slaver; Số dây truyền; Số lượng dữ liệu truyền; Pha và cực của clock; Tần số Clock; Chọn Slaver bằng phần cứng hay phần mềm; Truyền MSB hay LSB trước	Chọn cổng SPI; Chọn chế độ Master/Slaver; Số dây truyền; Kích thước dữ liệu truyền; Pha và cực của clock; Tần số Clock; Chọn Slaver bằng phần cứng hay phần mềm; Truyền MSB hay LSB trước	2
207	Các hàm trong thư viện HAL cho cổng SPI của STM32:		HAL_SPI_Init; HAL_SPI_DeInit; HAL_SPI_Transmit; HAL_SPI_Receive HAL_SPI_TransmitReceive HAL_SPI_GetStat HAL_SPI_GetError	HAL_SPI_Init; HAL_SPI_Config; HAL_SPI_Transmit; HAL_SPI_Receive HAL_SPI_TransmitReceive HAL_SPI_GetState HAL_SPI_GetError	HAL_SPI_Init; HAL_SPI_DeInit; HAL_SPI_Transmit; HAL_SPI_Receive HAL_SPI_Transceive HAL_SPI_GetState HAL_SPI_GetError	HAL_SPI_Init; HAL_SPI_Start; HAL_SPI_Transmit; HAL_SPI_Receive HAL_SPI_TransmitReceive HAL_SPI_GetState HAL_SPI_GetError	2

208	 <p>Trong sơ đồ kết nối, dữ liệu từ Master cung cấp tới chân MOSI sẽ:</p>		Được tất cả các Slaver nhận và xử lý	Được Slaver nhận được tín hiệu SS=0 nhận	Được Slaver nhận được tín hiệu SS=1 nhận	Slave có địa chỉ đúng với địa chỉ Master cung cấp nhận	2
209	Theo chuẩn SPI dữ liệu có thể được truyền theo chế độ:		Đơn công chỉ từ Master tới Slaver	Đơn công chỉ từ Slaver tới Master	Song công	Cả ba câu kia đều đúng	2
210	Khi nhận dữ liệu truyền tới từ Master, Slaver trong truyền nhận SPI đồng bộ với Master bằng cách nào?		Nhận dữ liệu ở cạnh lên hoặc xuống của tín hiệu Clock mà nó nhận từ Master	Kiểm tra trạng thay đổi logic của đường MOSI	Bắt đầu nhận dữ liệu khi SS=0	Đồng bộ bằng cách so sánh từ đồng bộ tới từ Master	2
211	Khi truyền dữ liệu tới Master, Slaver trong truyền nhận SPI đồng bộ với Master bằng cách nào?		Truyền dữ liệu ở cạnh lên hoặc xuống của tín hiệu Clock mà nó nhận từ Master	Kiểm tra trạng thay đổi logic của đường MOSI	Bắt đầu nhận dữ liệu khi SS=0	Đồng bộ bằng cách so sánh từ đồng bộ tới từ Master	2
212	Các tham số khởi động cho I2C của STM32 là:		Tốc độ, chu kỳ nhiệm vụ của clock; Chế độ địa chỉ; Địa chỉ Master; Chống nhiễu; Gọi tổng quát; Thời gian	Tốc độ, chu kỳ nhiệm vụ của clock; Chế độ địa chỉ; Địa chỉ Slaver; Chống nhiễu; Gọi riêng; Thời gian tăng	Tốc độ, chu kỳ nhiệm vụ của clock; Chế độ địa chỉ; Địa chỉ Slaver; Chống nhiễu; Gọi tổng quát; Thời gian	Tốc độ, pha của clock; Chế độ địa chỉ; Địa chỉ Slaver; Chống nhiễu; Gọi tổng quát; Thời gian tăng trên SDA;	2

			tăng trên SDA; Chế độ nhanh; Gửi bit ACK	trên SDA; Chế độ nhanh; Gửi bit ACK	tăng trên SDA; Chế độ nhanh; Gửi bit ACK	Chế độ nhanh; Gửi bit ACK	
213	Lệnh I2C1->OAR1 = (0x50 << 1) viết cho STM32 sẽ khởi động:		Khởi I2C1 là Slaver có địa chỉ là 80	Khởi I2C1 là Slaver có địa chỉ là 50	Khởi I2C1 là Master có địa chỉ là 80	Khởi I2C1 là Master có địa chỉ là 50	2
214	Các hàm trong thư viện HAL cho công I2C của STM32:		HAL_I2C_Init; HAL_I2C_DeInit HAL_I2C_Master _Trans; HAL_I2C_Master _Rece; HAL_I2C_Slave_ Trans; HAL_I2C_Slave _Rece; HAL_I2C_Stop()	HAL_I2C_Init; HAL_I2C_DeInit HAL_I2C_Master _Transmit; HAL_I2C_Master _Receive; HAL_I2C_Slave_ Transmit; HAL_I2C_Slave_ Receive; HAL_I2C_Stop()	HAL_I2C_Init; HAL_I2C_DeInit HAL_I2C_MOSI HAL_I2C_MISO; HAL_I2C_SOMI; HAL_I2C_SIMO; HAL_I2C_Stop()	HAL_I2C_Init; HAL_I2C_DeInit HAL_I2C_Master _Transeive; HAL_I2C_Slave_ Transceive; HAL_I2C_Stop()	2
215	Khi nhận dữ liệu truyền tới từ Master, Slaver trong truyền nhận I2C đồng bộ với Master bằng cách nào?		Nhận dữ liệu ở cạnh lên hoặc xuống của tín hiệu Clock mà nó nhận từ Master	Kiểm tra trạng thay đổi logic của đường SDA	Bắt đầu nhận dữ liệu khi tín hiệu đồng bộ tích cực	Đồng bộ bằng cách so sánh từ đồng bộ tới từ Master	2
216	Khi truyền dữ liệu tới Master, Slaver trong truyền nhận I2C đồng bộ với Master bằng cách nào?		Truyền dữ liệu ở cạnh lên hoặc xuống của tín hiệu Clock mà nó nhận từ Master	Kiểm tra trạng thay đổi logic của đường SDA	Bắt đầu nhận dữ liệu khi tín hiệu đồng bộ tích cực	Đồng bộ bằng cách so sánh từ đồng bộ tới từ Master	2

217	 <p>Trong sơ đồ kết nối trên, dữ liệu từ Master cung cấp tới chân SDA sẽ:</p>		Được tất cả các Slaver nhận và xử lý	Được Slaver mà Master chỉ định trước nhận	Được Slaver có địa chỉ bằng 1 nhận	Slave có địa chỉ đúng với địa chỉ Master cung cấp nhận	2
218	Theo chuẩn I2C dữ liệu có thể được truyền theo chế độ:		Đơn công chỉ từ Master tới Slaver	Đơn công chỉ từ Slaver tới Master	Song công	Bán song công giữa Master và Slaver	2
219	Các tham số cần khởi động cho USART của STM32 là:		Baud Rate; RE; TE; Word Length; Parity; Start bits; Oversampling; Buffer Size; Error Checking; Flow Control; GPIO Configuration; DMA; Syn/Asyn Mode	Baud Rate; RE; TE; Word Length; Parity; Stop bits; Oversampling; Buffer Size; Error Checking; Flow Control; GPIO Configuration; DMA; Syn/Asyn Mode	Baud Rate; RE; TE; Word Length; Parity; Stop bits; Oversampling; Buffer Enable/Disable; Error Checking; Flow Control; GPIO Configuration; DMA; Syn/Asyn Mode	Baud Rate; RE; TE; Word Length; Parity; Stop bits; Oversampling; Buffer Size; Error Checking; Flow Control; GPIO In/Out; DMA; Syn/Asyn Mode	2
220	Các hàm trong thư viện HAL cho cổng USART của STM32:		HAL_UART_Init; HAL_UART_DeInit; HAL_UART_Transmit; HAL_UART_Receive; HAL_UART_Transmit_IT; HAL_UART_Receive_IT; HAL_UART_Transmit_DMA; HAL_UART_Receive_DMA;	HAL_UART_Init; HAL_UART_DeInit; HAL_UART_Transmit; HAL_UART_Receive; HAL_UART_Transmit_IT; HAL_UART_Receive_IT; HAL_UART_Transmit_DMA; HAL_UART_Receive_DMA;	HAL_UART_Init; HAL_UART_Syn; HAL_UART_Transmit; HAL_UART_Receive; HAL_UART_Transmit_IT; HAL_UART_Receive_IT; HAL_UART_Transmit_DMA; HAL_UART_Receive_DMA; HAL_UART_IRQHandler;	HAL_UART_Init; HAL_UART_Asyn; HAL_UART_Transmit; HAL_UART_Receive; HAL_UART_Transmit_IT; HAL_UART_Receive_IT; HAL_UART_Transmit_DMA; HAL_UART_Receive_DMA;	2

			HAL_UART_IRQHandler;	HAL_UART_IRQHandler;		HAL_UART_IRQHandler;	
221	Khi nhận dữ liệu trong truyền nhận USART chế độ Asynchronous, bên nhận đồng bộ với bên truyền bằng cách nào?		Nhận dữ liệu ở cạnh lên hoặc xuống của tín hiệu SCK mà nó nhận từ bên truyền	Kiểm tra trạng thái thay đổi logic của đường TX	Bắt đầu nhận dữ liệu khi tín hiệu đồng bộ tích cực	Đồng bộ bằng cách so sánh từ đồng bộ tới từ bên truyền	2
222	Khi nhận dữ liệu trong truyền nhận USART chế độ Synchronous, bên nhận đồng bộ với bên truyền bằng cách nào?		Nhận dữ liệu ở cạnh lên hoặc xuống của tín hiệu SCK mà nó nhận từ bên truyền	Kiểm tra trạng thái thay đổi logic của đường TX	Bắt đầu nhận dữ liệu khi tín hiệu đồng bộ tích cực	Đồng bộ bằng cách so sánh từ đồng bộ tới từ bên truyền	2
223	Các lợi điểm khi sử dụng truyền USART ở chế độ DMA là:		Dữ liệu được truyền và nhận mà không cần can thiệp của CPU	Cho phép bạn truyền và nhận dữ liệu liên tục mà không cần phải cài đặt từng byte hoặc từng khung dữ liệu một	Đảm bảo rằng dữ liệu được truyền và nhận đúng lúc mà không bị ngắt quãng bởi xử lý CPU	Cả ba câu kia đều đúng	2
224	Các lợi điểm khi sử dụng ngắt nhận USART là:		CPU có thể thực hiện các công việc khác khi không có dữ liệu truyền tới	CPU sẽ nhận được dữ liệu nhanh hơn	Tốc độ truyền dữ liệu cao hơn	CPU không cần điều khiển việc nhận dữ liệu	2
225	<pre>#include "Device/Include/stm32f10x.h" void gpio_init(void) { RCC->APB2ENR= (1<<3)</pre>		Nhấp nháy 3 led chân nối Anode vào các chân PB7, PB8, PB9	Nhấp nháy 3 led chân nối Kathode vào các chân PB7, PB8, PB9	Nhấp nháy led chân nối Anode vào chân PB1	Nhấp nháy led chân nối Kathode vào chân PB1	3

	<pre> GPIOB->CRL= (3<<28); GPIOB->CRH=(3<<0) (3<<4); } void delay(volatile uint32_t a) { a=a*4000; while(a--); } int main(void) { gpio_init(); while(1) { GPIOB->ODR= (1<<7) (1<<8) (1<<9); delay(500); GPIOB->ODR&= ~((1<<7) (1<<8) (1<<9)); delay(500); } } </pre> <p>Chương trình viết dưới lớp thanh ghi của STM32 thực hiện:</p>						
226	<pre> #include "Device/Include/stm32f10x.h void gpio_init(void) { RCC->APB2ENR= (1<<3); GPIOB->CRL= (3<<20) (3<<24) (3<<28); GPIOB->CRH= (3<<0) (3<<4) (3<<8) (3<<12); } void delay(volatile uint32_t a) { a=a*4000; while(a--); } int main(void) { gpio_init(); while(1) { uint8_t i=0; </pre>		<p>Điều khiển các LED nối Kathode vào cổng B tắt dần</p>	<p>Điều khiển các LED nối Anode vào cổng B tắt dần</p>	<p>Điều khiển các LED nối Anode vào cổng B sáng dần</p>	<p>Điều khiển các LED nối Kathode vào cổng B sáng dần</p>	3

	<pre> for(i=5;i<=12;i++) { GPIOB->ODR= (1<<i); delay(500); GPIOB->ODR&= ~(1<<i); delay(500); } } </pre> <p>Chương trình viết dưới lớp thanh ghi của STM32 thực hiện:</p>						
227	Vùng nhớ Flash của STM32 Cortex có địa chỉ từ 0x08000000 tới 0x0801FFFF sẽ có dung lượng là:		16KB	32KB	64KB	128KB	3
228	<pre> #include "stm32f4xx_hal.h" #define LED_PIN GPIO_PIN_13 #define LED_PORT GPIOC #define BUTTON_PIN GPIO_PIN_0 #define BUTTON_PORT GPIOA int main(void) { HAL_Init(); -- HAL_RCC_GPIOC_CLK_ENABLE(); -- HAL_RCC_GPIOA_CLK_ENABLE(); GPIO_InitTypeDef LED_InitStruct; LED_InitStruct.Pin = LED_PIN; LED_InitStruct.Mode = GPIO_MODE_OUTPUT_PP; LED_InitStruct.Speed = GPIO_SPEED_FREQ_LOW; </pre>		Điều khiển LED sáng khi nhấn nút	Chuyển trạng thái sáng tắt của LED khi nút nhấn tác động	Điều khiển LED sáng khi không nhấn nút	Điều khiển LED chớp tắt khi nhấn nút	3

	<pre> HAL_GPIO_Init(LED_PORT, &LED_InitStruct); GPIO_InitTypeDef BUTTON_InitStruct; BUTTON_InitStruct.Pin = BUTTON_PIN; BUTTON_InitStruct.Mode = GPIO_MODE_INPUT; BUTTON_InitStruct.Pull = GPIO_PULLUP; HAL_GPIO_Init(BUTTON_PORT, &BUTTON_InitStruct); while (1) { GPIO_PinState buttonState = HAL_GPIO_ReadPin(BUTTON_P ORT, BUTTON_PIN); if (buttonState == GPIO_PIN_RESET) { HAL_GPIO_TogglePin(LED_POR T, LED_PIN); HAL_Delay(100); } } </pre> <p>Chương trình viết dưới lớp thư viện HAL cho STM32 thực hiện:</p>					
229	<pre> #include "stm32f4xx_hal.h" #define SEGMENT_PORT GPIOA #define DIGIT_PORT GPIOB const uint8_t digitValues[10] = { 0x3F,0x06, 0x5C,0x4F,0x66, ,0x6D,0x7D,0x07, 0x7F,0x6F }; void displayDigit(uint8_t digit) { HAL_GPIO_WritePin(DIGIT_POR T, DIGIT_1_PIN DIGIT_2_PIN DIGIT_3_PIN DIGIT _4_PIN, GPIO_PIN_RESET); switch (digit) { case 1: HAL GPIO WritePin(DIGIT POR </pre>	Sáng các số từ 0 đến 9 lên 1 LED 7 đoạn nối vào GPIOA	Sáng lần lượt các số từ 0 đến 9 lên 4 LED 7 đoạn các đoạn nối vào GPIOA, các tín hiệu chọn LED nối vào GPIOB	Sáng lần lượt các số từ 0 đến 9 lên 4 LED 7 đoạn các đoạn nối vào GPIOB, các tín hiệu chọn LED nối vào GPIOA	Sáng lần lượt các số từ 0 đến 3 lên 4 LED 7 đoạn các đoạn nối vào GPIOA, các tín hiệu chọn LED nối vào GPIOB	3

<pre> T, DIGIT_1_PIN, GPIO_PIN_SET); break; case 2: HAL_GPIO_WritePin(DIGIT_PORT, DIGIT_2_PIN, GPIO_PIN_SET); break; case 3: HAL_GPIO_WritePin(DIGIT_PORT, DIGIT_3_PIN, GPIO_PIN_SET); break; case 4: HAL_GPIO_WritePin(DIGIT_PORT, DIGIT_4_PIN, GPIO_PIN_SET); break; } } int main(void) { HAL_Init(); __HAL_RCC_GPIOA_CLK_ENABLE(); __HAL_RCC_GPIOB_CLK_ENABLE(); GPIO_InitTypeDef GPIO_InitStruct GPIO_InitStruct.Pin = SEGMENT_A_PIN SEGMENT_B_PIN SEGMENT_C_PIN SEGMENT_D_PIN SEGMENT_E_PIN SEGMENT_F_PIN SEGMENT_G_PIN DIGIT_1_PIN DIGIT_2_PIN DIGIT_3_PIN DIGIT_4_PIN; GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP; GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW; HAL_GPIO_Init(SEGMENT_PORT, &GPIO_InitStruct); while (1) { for (uint8_t digit = 1; digit <= 4; digit++) { for (uint8_t number = 0; number < 10; number++) { </pre>						
---	--	--	--	--	--	--

	<pre>HAL_GPIO_WritePin(SEGMENT_PORT, SEGMENT_A_PIN SEGMENT_B_PIN SEGMENT_C_PIN SEGMENT_D_PIN SEGMENT_E_PIN SEGMENT_F_PIN SEGMENT_G_PIN, digitValues[number]); displayDigit(digit); HAL_Delay(1000); } } }</pre> <p>Chương trình viết dưới lớp thư viện HAL cho STM32 thực hiện:</p>						
230	<pre>void LCD_W(uint8_t data) { HAL_GPIO_WritePin(RS_PORT, RS_PIN, GPIO_PIN_SET); LCD_Send4Bit(data >> 4); HAL_GPIO_WritePin(EN_PORT, EN_PIN, GPIO_PIN_SET); delay(1); HAL_GPIO_WritePin(EN_PORT, EN_PIN, GPIO_PIN_RESET); LCD_Send4Bit(data & 0x0F); HAL_GPIO_WritePin(EN_PORT, EN_PIN, GPIO_PIN_SET); delay(1); HAL_GPIO_WritePin(EN_PORT, EN_PIN, GPIO_PIN_RESET); delay(2); }</pre> <p>Hàm LCD_W cho STM32 dưới lớp thư viện HAL ở trên thực hiện:</p>		Ghi dữ liệu 8 bit tới LCD bằng hai lần 4 bit, 4 bit cao trước, 4 bit thấp sau	Ghi dữ liệu 8 bit tới LCD bằng hai lần 4 bit, 4 bit thấp trước, 4 bit cao sau	Ghi dữ liệu 8 bit tới LCD bằng một lần ghi	Cả ba câu kia đều đúng	3
231	<pre>void scanKeyboard(void) { for (int i = 0; i < ROWS; i++) { HAL_GPIO_WritePin(row_ports[i], row_pins[i], GPIO_PIN_RESET); for (int j = 0; j < COLS; j++) {</pre>		ROWS và COLS có giá trị từ 1 đến 4	ROWS và COLS có 1 trong các giá trị nhị phân từ 0000B đến 1111B	ROWS và COLS có giá trị từ 0 đến 3	ROWS và COLS có 1 trong các giá trị nhị phân từ 0001B;	3

	<pre> if HAL_GPIO_ReadPin (col_ports[j], col_pins[j]) == GPIO_PIN_RESET) { } HAL_GPIO_WritePin(row_ports [i], row_pins[i], GPIO_PIN_SET); } } </pre> <p>Sau khi gọi hàm quét bàn phím ma trận viết dưới lớp thư viện HAL cho STM32 ở trên sẽ được:</p>					0010B; 0100B; 1000B	
232	<pre> ADC_HandleTypeDef hadc1; hadc1.Instance = ADC1; hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV4; hadc1.Init.Resolution = ADC_RESOLUTION_12B; hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT; hadc1.Init.ScanConvMode = DISABLE; hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV; hadc1.Init.ContinuousConvMode = DISABLE; hadc1.Init.NbrOfConversion = 1; if (HAL_ADC_Init(&hadc1) != HAL_OK) { Error_Handler(); } </pre> <p>Đoạn chương trình STM32 dưới lớp thư viện HAL trên thực hiện:</p>		<p>Khởi động khối ADC1; chạy 1 kênh biến đổi; Chế độ biến đổi single mode; Độ phân giải 12 bit; Dữ liệu được chỉnh về bên phải của thanh ghi dữ liệu</p>	<p>Khởi động khối ADC1; chạy 16 kênh biến đổi; Chế độ biến đổi single mode; Độ phân giải 12 bit; Dữ liệu được chỉnh về bên phải của thanh ghi dữ liệu</p>	<p>Khởi động khối ADC1; chạy 1 kênh biến đổi; Chế độ biến đổi liên tục; Độ phân giải 12 bit; Dữ liệu được chỉnh về bên phải của thanh ghi dữ liệu</p>	<p>Khởi động khối ADC1; chạy 1 kênh biến đổi; Chế độ biến đổi single mode; Độ phân giải 16 bit; Dữ liệu được chỉnh về bên phải của thanh ghi dữ liệu</p>	3
233	<pre> #include <stdio.h> #define NUM_ROWS 4 #define NUM_COLS 4 </pre>		4	1	Ký tự A	0x0A	3

	<pre> keypad[NUM_ROWS][NUM_COLS] = { {1, 2, 3, 'A'}, {4, 5, 6, 'B'}, {7, 8, 9, 'C'}, {'*', 0, '#', 'D'} }; char readKeypad(int col, int row) { return keypad[row][col]; } int main() { int col = 4; int row = 1; char key = readKeypad(col, row); } </pre> <p>Sau đoạn chương trình trên sẽ được biến “key” bằng</p>						
234	<pre> #include <stdio.h> int main() { int num = 1642; char str[10]; snprintf(str, sizeof(str), "%d", num); printf("Result: %s\n", str); return 0; } </pre> <p>Sau khi thực hiện chương trình C trên biến str có giá trị là:</p>		Str={0x01,0x06,0x04,0x02}	Str={0x21,0x26,0x24,0x22}	Str={0x31,0x36,0x34,0x32}	Str={0x41,0x46,0x44,0x42}	3
235	Một ADC 10 bit có điện áp đầy thang là 3.3V đọc điện áp từ một cảm biến nhiệt độ có độ nhạy		24.748 °C	34.748 °C	44.748 °C	54.748 °C	3

	10mV/°C, điện áp tại 0°C là 0.5V. Giá trị đọc được là 123 sẽ tương ứng với nhiệt độ là:						
236	<pre>#include "stm32f4xx_hal.h" void Timer_Config(void) { TIM_HandleTypeDef htim; --HAL_RCC_TIM2_CLK _ENABLE(); htim.Instance = TIM2; htim.Init.Prescaler = HAL_RCC_GetPCLK1Freq() / 1000 - 1; htim.Init.CounterMode = TIM_COUNTERMODE_UP; htim.Init.Period=1000 - 1; htim.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1; HAL_TIM_Base_Init(&htim); HAL_NVIC_SetPriority(TIM2_IR Qn, 0, 0); HAL_NVIC_EnableIRQ(TIM2_I RQn); HAL_TIM_Base_Start_IT(&htim) ; } HAL_TIM_PeriodElapsedCallbac k(TIM_HandleTypeDef *htim) { if (htim->Instance == TIM2) { HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5); } } int main(void) { HAL_Init(); Timer_Config(); --HAL_RCC_GPIOA_ CLK _ENABLE();</pre>	Thay đổi trạng thái chân 5 GPIOA sau mỗi khoảng thời gian 1/10 giây bằng quét vòng	Thay đổi trạng thái chân 5 GPIOA sau mỗi khoảng thời gian 1/10 giây bằng chương trình ngắt	Thay đổi trạng thái chân 5 GPIOA sau mỗi khoảng thời gian 1 giây bằng quét vòng	Thay đổi trạng thái chân 5 GPIOA sau mỗi khoảng thời gian 1 giây bằng chương trình ngắt	3	

	<pre> GPIO_InitTypeDef GPIO_InitStruct = {0}; GPIO_InitStruct.Pin = GPIO_PIN_5; GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP; GPIO_InitStruct.Pull = GPIO_NOPULL; HAL_GPIO_Init(GPIOA, &GPIO_InitStruct); while (1) { } </pre> <p>Chương trình STM32 dưới lớp thư viện HAL ở trên thực hiện:</p>						
237	<pre> #include "stm32f4xx_hal.h" TIM_HandleTypeDef htim; void Timer_PWM_Config(void) { TIM_OC_InitTypeDef sConfigOC; --HAL_RCC_TIM2_CLK_ ENABLE(); htim.Instance = TIM2; htim.Init.Prescaler = 0; htim.Init.CounterMode = TIM_COUNTERMODE_UP; htim.Init.Period = 1000; htim.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1; HAL_TIM_PWM_Init(&htim); sConfigOC.OCMode = TIM_OCMode_PWM1; sConfigOC.Pulse = 500; sConfigOC.OCpolarity = TIM_OCPOLARITY_HIGH; sConfigOC.OCFastMode = TIM_OCFAST_DISABLE; HAL_TIM_PWM_ConfigChannel(</pre>		Cấu hình kênh 2 của Timer2 hoạt động ở chế độ Fast PWM với chu kỳ nhiệm vụ là 50%	Cấu hình kênh 1 của Timer2 hoạt động ở chế độ Fast PWM với mức cao là 1000 xung	Cấu hình kênh 1 của Timer2 hoạt động ở chế độ PWM mode 1 với chu kỳ nhiệm vụ là 50%	Cấu hình kênh 1 của Timer2 hoạt động ở chế độ Fast PWM chu kỳ là 500 xung	3

	<pre> &htim, &sConfigOC, TIM_CHANNEL_1); HAL_TIM_PWM_Start(&htim, TIM_CHANNEL_1); } int main(void) { HAL_Init(); Timer_PWM_Config(); while (1) { } } </pre> <p>Chương trình STM32 dưới lớp thư viện HAL ở trên thực hiện:</p>						
238	<pre> #include "stm32f4xx_hal.h" void EXTI_Config(void) { --HAL_RCC_GPIOA_ CLK_ENABLE(); GPIO_InitTypeDef GPIO_InitStruct = {0}; GPIO_InitStruct.Pin = GPIO_PIN_0; GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING; GPIO_InitStruct.Pull = GPIO_NOPULL; HAL_GPIO_Init(GPIOA, &GPIO_InitStruct); HAL_NVIC_SetPriority(EXTI0_ IRQn, 0, 0); HAL_NVIC_EnableIRQ(EXTI0_ IRQn); } void HAL_GPIO_EXTI_Callback(uin t16_t GPIO_Pin) { if (GPIO_Pin == GPIO_PIN_0) { } } </pre>		Có cạnh xuống ở chân 0 cổng GPIOA	Có cạnh lên ở chân 0 cổng GPIOA	Có mức thấp ở chân 0 cổng GPIOA	Có mức cao ở chân 0 cổng GPIOA	3

	<pre>int main(void) { HAL_Init(); EXTI_Config(); while (1) { } } } </pre> <p>Chương trình STM32 dưới lớp thư viện HAL ở trên chương trình ngắt sẽ thực hiện khi:</p>						
239	<pre>static void MX_TIM2_Init(void) { htim2.Instance = TIM2; htim2.Init.Prescaler = 0; htim2.Init.CounterMode = TIM_COUNTERMODE_UP; htim2.Init.Period = 65535; htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1; HAL_TIM_OnePulse_Init(&htim2, TIM_OPMODE_SINGLE); TIM_OC_InitTypeDef sConfigOC; sConfigOC.OCMode = TIM_OCMODE_PWM1; sConfigOC.OCpolarity = TIM_OCPOLARITY_HIGH; sConfigOC.Pulse = 32767; sConfigOC.OCNPolarity = TIM_OCNPOLARITY_HIGH; sConfigOC.OCFastMode = TIM_OCFAST_DISABLE; sConfigOC.OCIdleState = TIM_OCIDLESTATE_RESET; sConfigOC.OCNIdleState = TIM_OCNIDLESTATE_RESET; HAL_TIM_OnePulse_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_1); } </pre>		<p>Khởi động Timer2 hoạt động ở chế độ encoder</p>	<p>Khởi động Timer2 hoạt động ở chế độ đếm sự kiện ở kênh số 1</p>	<p>Khởi động Timer2 hoạt động ở chế độ one -pulse tạo ra xung ở kênh PWM1</p>	<p>Khởi động Timer2 hoạt động ở chế độ tạo ra xung PWM ở kênh PWM1</p>	3

	Hàm STM32 dưới lớp thư viện HAL ở trên thực hiện:						
240	<pre>#include "stm32f4xx_hal.h" TIM_HandleTypeDef htim2; void SystemClock_Config(void); static void MX_GPIO_Init(void); static void MX_TIM2_Init(void); int main(void) { HAL_Init(); SystemClock_Config(); MX_GPIO_Init(); MX_TIM2_Init(); HAL_TIM_Encoder_Start(&htim2, TIM_CHANNEL_1); HAL_TIM_Encoder_Start(&htim2, TIM_CHANNEL_2); while (1) { int encoderValue = __HAL_TIM_GET_COUNTER(&htim2); HAL_Delay(100); } void SystemClock_Config(void) { // Cấu hình hệ thống clock // ... } static void MX_TIM2_Init(void) { htim2.Instance = TIM2; htim2.Init.Prescaler = 0;</pre>	<p>Cấu hình Timer2 ở chế độ Encoder; Khởi động đọc giá trị Encoder ở kênh 1 và 2; Lấy giá trị Encoder ở chương trình chính</p>	<p>Cấu hình Timer2 ở chế độ Encoder; Khởi động đọc giá trị Encoder ở kênh 1; Lấy giá trị Encoder ở chương trình chính</p>	<p>Cấu hình Timer2 ở chế độ Encoder; Khởi động đọc giá trị Encoder ở kênh 2; Lấy giá trị Encoder ở chương trình chính</p>	<p>Cả ba câu kia đều đúng</p>	3	

	<pre> htim2.Init.CounterMode = TIM_COUNTERMODE_ENCODERMODE2; // Chọn chế độ encoder htim2.Init.Period = 65535; // Giá trị tối đa của bộ đếm htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV 1; HAL_TIM_Encoder_Init(&hti m2); } static void MX_GPIO_Init(void) { } </pre> <p>Chương trình STM32 dưới lớp thư viện HAL ở trên thực hiện:</p>						
241	Các lệnh nào sau đây của ARM thực hiện việc rẽ nhánh chương trình?		B BL	BX BLX	BXJ	Cả ba câu kia đều đúng	1
242	Trong các lệnh chế độ ARM, mã điều kiện sẽ chiếm độ dài là:		2 bit	4 bit	6 bit	8 bit	1
243	Trong các lệnh chế độ THUMB của ARM, mã điều kiện sẽ chiếm độ dài là:		2	4	6	Không có mã điều kiện trong lệnh	1
244	Mã điều kiện trong các lệnh ARM được sử dụng để làm gì?		Xét điều kiện sau khi thực hiện lệnh	Xác định lệnh có được thực hiện không	Quản lý các điều kiện của chương trình	Quản lý biến số trong chương trình	1
245	ARM7 có bao nhiêu mã điều kiện?		10	15	12	17	1

246	Mã điều kiện trong một lệnh của ARM khi xét là của:		Lệnh không có hậu tố S vừa thực hiện trước đó	Sau khi thực hiện dòng lệnh	Của một lệnh số học	Của một lệnh logic	1
247	Khi lệnh có mã điều kiện HS nó sẽ thực hiện khi:		Kết quả lệnh so sánh trước đó là nhỏ hơn hoặc bằng	Kết quả lệnh so sánh trước đó là lớn hơn	Kết quả lệnh so sánh trước đó là lớn hơn hoặc bằng	Kết quả lệnh so sánh trước đó là nhỏ hơn	1
248	Điều kiện "EQ" trong lệnh ARM dùng để kiểm tra điều gì?		Bằng nhau	Lớn hơn	Lớn hơn hoặc bằng	Không bằng	1
249	Điều kiện "NE" trong lệnh ARM dùng để kiểm tra điều gì?		Bằng nhau	Lớn hơn	Lớn hơn hoặc bằng	Không bằng	1
250	Điều kiện "MI" trong lệnh ARM dùng để kiểm tra điều gì?		Tràn số	Số âm	Số dương	Tràn bit	1
251	Điều kiện "PL" trong lệnh ARM dùng để kiểm tra điều gì?		Lớn hơn hoặc bằng	Số âm	Nhỏ hơn hoặc bằng	Tràn bit	1
252	Điều kiện "VS" trong lệnh ARM dùng để kiểm tra điều gì?		Tràn	Không tràn	Nhớ	Không nhớ	1
253	Điều kiện "VC" trong lệnh ARM dùng để kiểm tra điều gì?		Tràn	Không tràn	Nhớ	Không nhớ	1
254	Điều kiện "HI" trong lệnh ARM dùng để kiểm tra điều gì?		Lớn hơn không tính dấu	Lớn hơn có tính dấu	Lớn hơn hoặc bằng có tính dấu	Lớn hơn hoặc bằng không tính dấu	1

255	Điều kiện "LS" trong lệnh ARM dùng để kiểm tra điều gì?		Nhỏ hơn không tính dấu	Nhỏ hơn có tính dấu	Nhỏ hơn hoặc bằng có tính dấu	Nhỏ hơn hoặc bằng không tính dấu	1
256	Điều kiện "GE" trong lệnh ARM dùng để kiểm tra điều gì?		Lớn hơn không tính dấu	Lớn hơn có tính dấu	Lớn hơn hoặc bằng có tính dấu	Lớn hơn hoặc bằng không tính dấu	1
257	Điều kiện "LE" trong lệnh ARM dùng để kiểm tra điều gì?		Nhỏ hơn không tính dấu	Nhỏ hơn có tính dấu	Nhỏ hơn hoặc bằng có tính dấu	Nhỏ hơn hoặc bằng không tính dấu	1
258	Trong mã điều kiện ARM, điều kiện "GT" có ý nghĩa gì?		Lớn hơn không tính dấu	Lớn hơn có tính dấu	Lớn hơn hoặc bằng có tính dấu	Lớn hơn hoặc bằng không tính dấu	1
259	Trong mã điều kiện ARM, điều kiện "AL" có ý nghĩa gì?		Lặp vô hạn	Luôn luôn đúng	Luôn không đúng	Cảnh báo	1
260	Tiền tố S trong một lệnh Parallel của ARM có ý nghĩa gì?		Duy trì dấu của kết quả phép tính chia dư	Kết quả phép tính chia dư sẽ luôn dương	Kết quả phép tính chia dư sẽ luôn âm	Không duy trì dấu của kết quả phép tính chia dư	1
261	Tiền tố Q trong một lệnh Arithmetic Parallel của ARM có ý nghĩa gì?		Chấp nhận kết quả phép tính số học có tràn	Đảm bảo kết quả của các phép tính nằm trong khoảng giá trị được xác định, thường là giá trị tối thiểu và tối đa mà kiểu dữ liệu có thể biểu diễn	Kết quả phép tính số học là bão hòa theo khả năng của thanh ghi	Cả ba câu kia đều đúng	1
262	Tiền tố U trong một lệnh Arithmetic Parallel của ARM có ý nghĩa gì?		Duy trì dấu của kết quả phép tính chia dư	Kết quả phép tính chia dư sẽ luôn dương	Kết quả phép tính chia dư sẽ luôn âm	Không duy trì dấu của kết quả phép tính chia dư	1

263	Tiền tố SH trong một lệnh Arithmetic Parallel của ARM có ý nghĩa gì?		Thực hiện các phép tính số học có dấu	Thực hiện các phép tính số học không dấu	Thực hiện các phép tính số học có dấu dương	Thực hiện các phép tính số học có dấu âm	1
264	Chế độ địa chỉ gián tiếp thanh ghi của ARM sử dụng thanh ghi để:		Chứa địa chỉ bộ nhớ cần truy cập	Chứa dữ liệu cần truy cập	Giữ độ dời địa chỉ dữ liệu cần truy cập	Giữ giá trị dịch của địa chỉ	1
265	Lệnh ldr r0, [r1] thực hiện:		Lấy giá trị trong r1 vào thanh ghi r0	Lấy giá trị trong ô nhớ có địa chỉ giữ trong r1 vào thanh ghi r0	Lấy giá trị trong r0 vào thanh ghi r1	Lấy giá trị trong ô nhớ có địa chỉ giữ trong r0 vào thanh ghi r1	2
266	Trong các lệnh của ARM toán hạng tức thời (immediate) được:		Giữ trong một ô nhớ dữ liệu	Giữ trong một thanh ghi	Mã hóa trong mã lệnh	Nằm trong ngăn xếp	2
267	Trong chế độ thanh ghi với độ dời tức thời (Register with immediate offset) của ARM địa chỉ của dữ liệu cần truy cập là:		Giá trị của thanh ghi cộng hoặc trừ đi một giá trị được mã hóa trong lệnh	Giá trị của một thanh ghi được nạp một giá trị mã hóa trong lệnh	Một giá trị được mã hóa trong lệnh	Không có địa chỉ trong chế độ này	2
268	Khi r1 = 1000, lệnh ldr r0, [r1, #8] thực hiện:		Lấy giá trị trong ô nhớ có địa chỉ 1000 vào thanh ghi r0 sau đó cộng với 8	Lấy giá trị trong ô nhớ có địa chỉ 1008 vào thanh ghi r0	Lấy giá trị 1008 vào thanh ghi r1	Lấy giá trị 1008 vào thanh ghi r0	2
269	Trong chế độ thanh ghi với độ dời thanh ghi (Register with register offset) của ARM địa chỉ của dữ liệu cần truy cập là:		Giá trị hai thanh ghi cộng hoặc trừ với nhau	Giá trị của thanh ghi cơ sở (base)	Giá trị của thanh ghi độ dời	Không có địa chỉ trong chế độ này	2

270	Lệnh ldr r0, [r1,r2] thực hiện:		Lấy giá trị r1 + r2 vào thanh ghi r0	Lấy giá trị trong ô nhớ có địa chỉ r1 + r2 vào thanh ghi r0	Ghi giá trị trong r0 vào địa chỉ r1 + r2	Lấy giá trị trong ô nhớ có địa chỉ giữ trong r1 vào r0 sau đó cộng với r2	2
271	Toán hạng LSL #10 thực hiện:		Nạp giá trị 10 vào thanh ghi LSL	Dịch phải logic thanh ghi độ dài 10 bit	Dịch trái logic thanh ghi độ dài 10 bit	Lấy giá trị trong ô nhớ có địa chỉ LSL + 10	2
272	Toán hạng LSR #10 thực hiện:		Nạp giá trị 10 vào thanh ghi LSR	Dịch phải logic thanh ghi độ dài 10 bit	Dịch trái logic thanh ghi độ dài 10 bit	Lấy giá trị trong ô nhớ có địa chỉ LSR + 10	2
273	Toán hạng ASR #10 thực hiện:		Nạp giá trị 10 vào thanh ghi ASR	Dịch phải số học thanh ghi độ dài đi 10 bit	Dịch phải logic thanh ghi độ dài 10 bit	Lấy giá trị trong ô nhớ có địa chỉ ASR + 10	2
274	Toán hạng ROR #10 thực hiện:		Nạp giá trị 10 vào thanh ghi ROR	Quay phải thanh ghi độ dài 10 bit	Dịch phải thanh ghi độ dài đi 10 bit	Lấy giá trị trong ô nhớ có địa chỉ ROR + 10	2
275	Toán hạng RRX thực hiện:		Quay phải dữ liệu 33 bit	Quay phải dữ liệu 32 bit	Quay phải dữ liệu 16 bit	Quay phải dữ liệu 8 bit	2
276	Chế độ địa chỉ thanh ghi với thanh ghi độ dài tỷ lệ (Register with scaled register offset), địa chỉ dữ liệu cần truy cập là:		Tổng thanh ghi cơ sở và thanh ghi độ dài đã được cộng thêm một giá trị	Tổng thanh ghi cơ sở và thanh ghi độ dài đã được chia tỷ lệ	Tổng thanh ghi cơ sở và thanh ghi độ dài đã được trừ đi một giá trị	Tổng giữa thanh ghi cơ sở với thanh ghi độ dài đã được quay dịch	2
277	Lệnh ldr r0, [r1, r2, LSL #2] thực hiện:		Lấy giá trị trong ô nhớ có địa chỉ [(r1+r2)<<2]	Lấy giá trị trong ô nhớ có địa chỉ [r1+(r2<<2)]	Lấy giá trị trong ô nhớ có địa chỉ [(r1+r2+LSL)<<2]	Lấy giá trị trong ô nhớ có địa chỉ [r1+r2+LSL<<2]	2
278	Lệnh ldr r0, [r1, -r2, ASR #2] thực hiện:		Lấy giá trị trong ô nhớ có địa chỉ [(r1-r2)>>2]	Lấy giá trị trong ô nhớ có địa chỉ [r1-(r2>>2)]	Lấy giá trị trong ô nhớ có địa chỉ [(r1-r2+ASL)<<2]	Lấy giá trị trong ô nhớ có địa chỉ [r1-r2+ASL<<2]	2

279	Chỉ thị chỉ số trước (Pre-Indexed) trong lệnh ARM thực hiện việc:		Thay đổi thanh ghi chứa địa chỉ bộ nhớ sau khi truy cập.	Thay đổi giá trị dữ liệu bộ nhớ trước khi truy cập.	Thay đổi thanh ghi chứa địa chỉ bộ nhớ trước khi truy cập.	Thay đổi giá trị dữ liệu bộ nhớ sau khi truy cập.	2
280	Lệnh ldr r0, [r1, #4]! thực hiện:		$r1 = r1 + 4$ $r0 = [r1]$	$r0 = [r1]$ $r1 = r1 + 4$	$r1 = r1 + 4$ $[r1] = r0$	$[r1] = r0$ $r1 = r1 + 4$	2
281	Lệnh ldr r0, [r1, r2, lsl #2]! thực hiện:		$r0 = [r1]$ $r1 = r1 + (r2 \ll 2)$	$r1 = (r1 + r2) \ll 2$ $r0 = [r1]$	$r1 = r1 + (r2 \ll 2)$ $r0 = [r1]$	$r0 = [r1]$ $r1 = (r1 + r2) \ll 2$	2
282	Chỉ thị chỉ số sau (Post-Indexed) trong lệnh ARM thực hiện việc:		Thay đổi thanh ghi chứa địa chỉ bộ nhớ sau khi truy cập.	Thay đổi giá trị dữ liệu bộ nhớ trước khi truy cập.	Thay đổi thanh ghi chứa địa chỉ bộ nhớ trước khi truy cập.	Thay đổi giá trị dữ liệu bộ nhớ sau khi truy cập.	2
283	Lệnh ldr r0, [r1], #4 thực hiện:		$r1 = r1 + 4$ $r0 = [r1]$	$r0 = [r1]$ $r1 = r1 + 4$	$r1 = r1 + 4$ $[r1] = r0$	$[r1] = r0$ $r1 = r1 + 4$	2
284	Lệnh ldr r0, [r1], r2, lsl #2 thực hiện:		$r0 = [r1]$ $r1 = r1 + (r2 \ll 2)$	$r1 = (r1 + r2) \ll 2$ $r0 = [r1]$	$r1 = r1 + (r2 \ll 2)$ $r0 = [r1]$	$r0 = [r1]$ $r1 = (r1 + r2) \ll 2$	2
285	Khi ARM sử dụng PC làm thanh ghi cơ sở truy cập bộ nhớ thì:		Có thể sử dụng Post – Index	Có thể sử dụng Pre – Index	Không thể sử dụng Post – Index và Pre - Index	Sử dụng Post – Index và Pre – Index đồng thời trong một lệnh	2
286	Lệnh ldr r3, [pc, #5] thực hiện:		Lấy giá trị trong ô nhớ có địa chỉ pc+5 vào r3	Lấy giá trị trong ô nhớ có địa chỉ pc vào r3 sau đó cộng với 5	Ghi giá trị của r3 vào ô nhớ có địa chỉ pc+5	Ghi giá trị của r3 + 5 vào ô nhớ có địa chỉ pc	2
287	Lệnh ldr Rd, =imm32 thực hiện:		Nạp hằng số chứa trong bộ nhớ chương trình vào thanh ghi	Nạp hằng số tức thời vào thanh ghi	Ghi giá trị tức thời vào bộ nhớ chương trình	Ghi giá trị tức thời vào vùng nhớ dữ liệu	2

288	Sau các lệnh: MOV R0, #42 MOV R1, #255 ADD R0, R0, R1 Thanh ghi R0 có giá trị là:		297	42	255	Không xác định	2
289	Sau các lệnh: MOV R0, #42 MOV R1, #255 ADD R0, R0, R1 Thanh ghi R1 có giá trị là:		297	42	255	Không xác định	2
290	Sau các lệnh: MOV R0, #42 MOV R1, #255 SUB R1, R1, R0 Thanh ghi R1 có giá trị là:		255	42	213	Không xác định	2
291	Sau các lệnh: MOV R0, #42 MOV R1, #255 SUB R1, R1, R0 Thanh ghi R0 có giá trị là:		255	42	213	Không xác định	2
292	Sau các lệnh: MOV R0, #42 MOV R1, #255 MVN R0, R1 Thanh ghi R0 có giá trị là:		0	42	255	Không xác định	2
293	Sau khi thực hiện lệnh: MOV R0, #42 TST R0, #10 Thanh ghi R0 có giá trị là:		10	0	42	Không xác định	2

294	Sau khi thực hiện lệnh: MOV R0, #42 TST R0, #10 Lệnh rẽ nhánh nào sẽ thực hiện rẽ nhánh chương trình:		BEQ	BNE	BVS	BLT	2
295	Sau khi thực hiện lệnh: MOV R0, #42 EOR R0, R0,R0 Thanh ghi R0 có giá trị là:		84	0	42	Không xác định	2
296	Sau khi thực hiện lệnh: MOV R0, #42 AND R0, R0,#0xFF Thanh ghi R0 có giá trị là:		84	0	42	Không xác định	2
297	Sau khi thực hiện lệnh: MOV R0, #42 AND R0, R0,#0 Thanh ghi R0 có giá trị là:		84	0	42	Không xác định	2
298	Sau khi thực hiện lệnh: MOV R0, #42 ORR R0, R0,#0xFF Thanh ghi R0 có giá trị là:		84	0XFF	42	Không xác định	2
299	Sau khi thực hiện lệnh: MOV R0, #42 ORR R0, R0,#0 Thanh ghi R0 có giá trị là:		84	0XFF	42	Không xác định	2
300	Lệnh STR R0, [R1] thực hiện:		Lưu giá trị thanh ghi R1 vào thanh ghi R0	Lưu giá trị thanh ghi R1 ô nhớ có địa chỉ giữ trong R0	Lưu giá trị thanh ghi R0 ô nhớ có địa chỉ giữ trong R1	Lấy giá trị trong ô nhớ có địa chỉ giữ trong R1 vào thanh ghi R0	2

301	<p>Sau khi thực hiện lệnh: MOV R0, #42 CMP R0, #10</p> <p>Lệnh rẽ nhánh nào sẽ thực hiện rẽ nhánh chương trình:</p>		BEQ	BGT	BVS	BLT	2
302	<p>foo: ADD R0, R1, R2 ; BX LR ;</p> <p>MAIN: MOV R1, #5 ; MOV R2, #3 ; BL foo</p> <p>Trong đoạn chương trình trên sau khi thực hiện lệnh “BX LR” điều khiển chương trình sẽ:</p>		Chuyển về sau lệnh “BL foo”	Chuyển về nhãn MAIN	Chuyển về nhãn foo	Chuyển tới lệnh “BL LR”	2
303	<p>.thumb foo: ADD R1, R1, #5 ; BX LR ; .code 32</p> <p>MAIN: MOV R2,#10; MOV R1, #12 ; BLX foo ; MOV R2,R1</p> <p>Sau khi thực hiện đoạn chương trình trên R2 có giá trị bằng bao nhiêu?</p>		#17	#15	#10	#5	2

304	Sau khi thực hiện lệnh: MOV R0, #42 EOR R0, R0,#0xFF Thanh ghi R0 có giá trị là:		84	24	42	00	2
305	AREA LoopExample, CODE, READONLY EXPORT main main MOV R0, #1 MOV R1, #10 loop_start ADD R0, R0, #1 SUBS R1, R1, #1 BNE loop_start SWI 0x11 END Sau khi thực hiện chương trình hợp ngữ ARM trên sẽ được:		R0=0; R1=10	R0=10; R1=0	R0=11; R1=0	R0=0; R1=11	3
306	ADR R4,a LDR R0,[R4] ADR R4,b LDR R1,[R4] ADD R3,R0,R1 ADR R4,c LDR R2,[R4] SUB R3,R3,R2 ADR R4,x STR R3,[R4]		x=a+b+c	x=a-b-c	x=a+b-c	x=-a+b-c	3

	Đoạn chương trình hợp ngữ ở trên tương đương khi viết trong C:						
307	ADR R4,b LDR R0,[R4] ADR R4,c LDR R1,[R4] ADD R2,R0,R1 ADR R4,a LDR R0,[R4] MUL R2,R2,R0 ADR R4,y STR R2,[R4] Đoạn chương trình hợp ngữ ở trên tương đương khi viết trong C:		$y=a*b+c$	$y=a+b*c$	$y=(a+b)*c$	$y=a*(b+c)$	3
308	ADR R4,a LDR R0,[R4] MOV R0,R0,LSL#2 ADR R4,b LDR R1,[R4] AND R1,R1,#15 ORR R1,R0,R1 ADR R4,z STR R1,[R4] Đoạn chương trình hợp ngữ ở trên tương đương khi viết trong C:		$z=(a<<2) (b\&15)$	$z=(a\&15) (b<<2)$	$z=(a (b\&15)<<2)$	$z=(a (b<<2)\&15)$	3
309	SUBS R1,R1,#0 ADDEQ R1, R1,#10		if	if ... else	switch ...case	while	3

	Các lệnh hợp ngữ trên tương đương với cấu trúc lập trình nào trong C?						
310	SUB R0,R0 START: CMP R0,#15 ADDLT R1,R1,R1 ADDLT R0,R0,#1 BLT START Đoạn chương trình hợp ngữ ARM trên tương đương với cấu trúc lập trình nào trong C?		if	if ... else	switch ...case	for	3
311	ADR R4,a LDR R0,[R4] MOV R0,R0,LSL#2 ADR R4,b LDR R1,[R4] CMP R0,R1 BGE fblock MOV R0,#5 ADR R4,x STR R0,[R4] ADR R4,c LDR R0,[R4] ADR R4,d LDR R1,[R4] ADD R0,R0,R1 ADR R4,y STR R0,[R4]		if (a>b) { x= c-d; } else; { x=5; y=c+d; }	if (a>b) { x=5; y=c+d; } else x= c-d;	if (a>b) { x=5; y=c+d; }	if (a>b) { x=c-d }	3

	B after fblock: ADR R4,c LDR R0,[R4] ADR R4,d LDR R1,[R4] SUB R0,R0,R1 ADR R4,x STR R0,[R4] after: ... Đoạn chương trình hợp ngữ ở trên tương đương khi viết trong C:						
312	ADR R4,n LDR R0,[R4] MLA R1,R0,R0,R0 MOV R0,R1,LSR#1 Đoạn chương trình hợp ngữ trên thực hiện công việc:		Tính $n*n*n$	Tính $n*n+n$ chia 2	Tính tổng n số tự nhiên đầu tiên	Tính $n*n$ chia 2	3
313	ADR R4,n LDR R0,[R4] MOV R1,#0 Loop: ADD R1,R1,R0 SUB R0,R0,#1 BNE Loop Đoạn chương trình hợp ngữ trên thực hiện công việc:		Tính $n*n*n$	Tính $n*n$	Tính tổng n số tự nhiên đầu tiên	Tính $n+n$ chia 2	3

314	<pre> MOV R0, #0 MOV R1, #5 MOV R2, #100 loop: CMP R1, R2 BGT done ADD R0, R0, R1 ADD R1, R1, #7 B loop done: MOV R7, #1 SWI 0 </pre> <p>Chương trình hợp ngữ ARM trên tương đương với cấu trúc lập trình nào trong C?</p>		while	do ... while	for	switch ...case	3
315	<pre> .data N .word 10 Sum .word 0 .text .global main main: LDR R0, =N MOV R1, #0 loop: ADD R1, R1, R0 SUB R0, R0, #1 CMP R0, #0 BNE loop </pre>		while	do ... while	for	switch ...case	3

	MOV R7, #1 SWI 0 Chương trình hợp ngữ ARM trên tương đương với cấu trúc lập trình nào trong C?						
316	<pre> .data Array .word 5, 2, 9, 1, 5, 6, 3 Length .word 7 .text .global main main: LDR R0, =Array LDR R1, =Length outer_loop: MOV R2, #0 MOV R3, #0 inner_loop: LDR R4, [R0, R3] LDR R5, [R0, R3, #4] CMP R4, R5 BLE no_swap STR R5, [R0, R3] STR R4, [R0, R3, #4] no_swap: ADD R3, R3, #4 CMP R3, R1 BLT inner_loop ADD R2, R2, #1 </pre>	Sắp xếp mảng dữ liệu Array theo giá trị từ nhỏ đến lớn theo thuật toán quick sort	Sắp xếp mảng dữ liệu Array theo giá trị từ lớn đến nhỏ theo thuật toán quick sort	Sắp xếp mảng dữ liệu Array theo giá trị từ nhỏ đến lớn theo thuật toán bubble sort	Sắp xếp mảng dữ liệu Array theo giá trị từ lớn đến nhỏ theo thuật toán bubble sort	3	

	<pre> CMP R2, R1 BLT outer_loop MOV R7, #1 SWI 0 </pre> <p>Chương trình hợp ngữ ARM trên thực hiện công việc:</p>						
317	<pre> .text .global main main: LDR R0, =TCode BXJ R0 B end TCode: @ đoạn code end: MOV R7, #1 SWI 0 </pre> <p>Trong chương trình ARM ở trên, đoạn lệnh sau nhãn “Tcode” là:</p>		Đoạn mã Java được chuyển đổi thành mã máy	Đoạn mã java nguyên gốc	Đoạn mã Java được chuyển đổi thành hợp ngữ	Đoạn lệnh ở trạng thái THUMB	3
318	<pre> .data Input .word 42 Result .word 0 Buffer .space 10 .text .global main main: LDR R0, =Input LDR R1, =Buffer </pre>		Chuyển đổi chuỗi số hệ 16 trong “Input” thành chuỗi số nguyên và lưu trữ trong bộ nhớ đệm “Buffer”. Sau đó, chương trình chuyển đổi chuỗi số nguyên	Chuyển đổi số nguyên trong “Input” thành chuỗi ký tự hệ 16 và lưu trữ trong bộ nhớ đệm “Result”. Sau đó, chương trình chuyển đổi chuỗi ký tự hệ 16 trở	Chuyển đổi số nguyên trong “Input” thành chuỗi ký tự hệ 16 và lưu trữ trong bộ nhớ đệm “Buffer”. Sau đó, chương trình chuyển đổi chuỗi ký tự hệ 16 trở lại thành số	Chuyển đổi chuỗi số hệ 16 trong “Input” thành chuỗi số nguyên và lưu trữ trong bộ nhớ đệm “Result”. Sau đó, chương trình chuyển đổi chuỗi số nguyên	3

	<pre> BL DecimalToHex LDR R1, =Buffer LDR R2, =Result BL HexToDecimal B end DecimalToHex: MOV R2, #0 loop1: MOV R3, R0, LSR #4 MUL R4, R3, #16 SUB R0, R0, R4 CMP R3, #9 ADDLS R3, R3, #'0' ADDHI R3, R3, #('A' - 10) STRB R3, [R1], #1 ADD R2, R2, #1 CMP R0, #0 BNE loop1 STRB R2, [R1], #1 MOV R7, #0 HexToDecimal: MOV R3, #0 loop2: LDRB R4, [R1], #1 CMP R4, #0 SUB R4, R4, #'0' ADD R3, R3, R4, LSL #4 </pre>		trở lại thành chuỗi ký tự hệ 16 và lưu trong biến “Result”	lại thành số nguyên và lưu trong biến “Buffer”	nguyên và lưu trong biến “Result”	trở lại thành chuỗi ký tự hệ 16 và lưu trong biến “Buffer”	
--	--	--	--	--	-----------------------------------	--	--

	<pre> B loop2 done2: STR R3, [R2] MOV R7, #0 end: MOV R7, #1 SWI 0 Chương trình hợp ngữ ARM trên thực hiện công việc: </pre>						
319	<pre> .data Array .word 5, 2, 9, 1, 5, 6, 3 Length .word 7 M1 .word 0 M2 .word 0 .text .global main main: LDR R0, =Array LDR R1, =Length LDR R2, =M1 LDR R3, =M2 MOV R4, #0 LDR R5, [R0, R4] STR R5, [R2] STR R5, [R3] loop: ADD R4, R4, #4 </pre>		<p>Tìm giá trị nhỏ nhất lưu vào M2 giá trị lớn nhất lưu vào M1</p>	<p>Tìm giá trị nhỏ nhất lưu vào M1 giá trị lớn nhất lưu vào M2</p>	<p>Tìm giá trị lớn nhất lưu vào M2</p>	<p>Tìm giá trị nhỏ nhất lưu vào M1</p>	3

	<pre> CMP R4, R1 BGE done LDR R5, [R0, R4] CMP R5, [R2] BLE not_max STR R5, [R2] not_max: CMP R5, [R3] BGE loop @ STR R5, [R3] B loop done: MOV R7, #1 SWI 0 </pre> <p>Chương trình hợp ngữ trên thực hiện công việc:</p>						
320	<pre> section .text global main main: ldr r0, =0x40023830 mov r1, #0x08 str r1, [r0] ldr r0, =0x40020C00 mov r1, #0x1000 str r1, [r0] loop: ldr r0, =0x40020C18 mov r1, #0x1000 str r1, [r0] </pre>	<p>Cấp Clock cho GPIOD; Khởi động GPIOD chân 12 là Input; Nhận dữ liệu từ GPIOD chân 12</p>	<p>Cấp Clock cho GPIOD; Khởi động GPIOD chân 12 là Output; Bật tắt LED gắn trên GPIOD chân 12</p>	<p>Cấp Clock cho GPIOD; Khởi động GPIOD chân 12 là Output; Bật LED gắn trên GPIOD chân 12</p>	<p>Cấp Clock cho GPIOD; Khởi động GPIOD chân 12 là Output; Cấp mức logic 0 ra GPIOD chân 12</p>	3	

	<pre>ldr r2, =1000000 delay_on: subs r2, #1 bne delay_on ldr r0, =0x40020C1C mov r1, #0x1000 str r1, [r0] ldr r2, =1000000 delay_off: subs r2, #1 bne delay_off b loop</pre> <p>Chương trình hợp ngữ trên thực hiện công việc:</p>						
--	--	--	--	--	--	--	--

GIẢNG VIÊN BIÊN SOẠN

(Ký, ghi rõ họ tên)

Phạm Thế Duy

