**CÂU HỎI THI TRẮC NGHIỆM**

**Tên học phần: Kỹ Thuật Vi Xử Lý** **Mã học phần:** **ELE1317**

**Số tín chỉ**: **03**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **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 quan trọng nhất giữa chúng là gì? |  | ROM lưu trữ dữ liệu dùng cho việc khởi động máy tính, trong khi RAM lưu trữ dữ liệu làm việc trong quá trình hoạt động của máy tính | ROM lưu trữ dữ liệu dùng cho các chương trình ứng dụng, trong khi RAM lưu trữ dữ liệu hệ điều hành | ROM lưu trữ dữ liệu mà người dùng nhập vào, trong khi RAM lưu trữ dữ liệu hệ thống | ROM lưu trữ dữ liệu lưu trữ tạm thời, trong khi RAM lưu trữ dữ liệu dùng cho việc khởi động máy tính | 1 |
| 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 ví 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 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 ví trí sẽ truyền dữ liệu với CPU | Cả ba câu kia đều đúng | 2 |
| 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 thưởng 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 đề 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 địa chỉ và 8 đường dữ liệu? |  | 2MB | 1MB | 4MB | 8MB | 2 |
| 20 | Điều gì xảy ra khi thanh ghi bộ đếm chương trình PC trỏ 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 bằng cách sử dụng khối chức năng nào? |  | Điện áp đầu vào | Điện áp nguồn | Timer/Counter | Điều khiển từ xa | 2 |
| 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 một thời điểm, mỗi lệnh thực hiện một công đoạn khác nhau | Là việc thực hiện tuần tự từng lệnh, lệnh này thực hiện xong mới tới lệnh kế tiếp | Là việc thực hiện các lệnh một cấu trúc ống trong vi xử lý | Cả ba câu kia đều đúng | 2 |
| 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 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 ghi 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 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 cấp dữ liệu ra data bus | 3 |
| 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 phần 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 tự 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 lệnh tiếp theo thực hiện là: |  | Được lấy từ đỉnh ngăn xếp | Giá trị hiện tại của PC cộng hoặc trừ một số độ dời mã hóa trong lệnh | Là một giá trị được mã hóa trong lệnh | Là giá trị của thanh ghi cờ | 1 |
| 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 báo 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, Undefine | 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? |  | Khối 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 trỏ tới lệnh viết kế tiếp trong chương trình | Cộng trừ đi một giá trị lưu trong mã lệnh để trỏ tới một vị trí lấy lệnh mới | Nạp một giá trị mới lưu trong mã lệnh để trỏ 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 trỏ 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 phần cứng | Chuyển điều khiển bằng ngắt cứng được tác động bằng tín hiệu điện phần 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 phần cứng 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 đỉnh 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 đỉnh 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, chia, AND, OR, XOR, NOT thì nó cần nhận được ít nhất bao nhiêu tín hiệu điều khiển từ CU? |  | 8 | 3 | 4 | 7 | 3 |
| 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; Cấp dữ liệu ra data BUS | 1 |
| 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 dừng 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 động bộ cung cấp từ CPU | Tín hiệu điều khiển động 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 phần cứng đơn giản | Có cấu trúc đường ống cho phép thực hiện nhiều lệnh tại cùng một thời điểm | 2 |
| 134 | Trong định thời bộ nhớ SRAM ở trên, tín hiệu OE là: |  | 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 |
| 135 | Trong định thời bộ nhớ ROM ở trên, khi CE ở mức cao: |  | 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 | Trong định thời (Timing diagram) ở trên, để thực hiện chu kỳ đọc dữ liệu cần: |  | 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 | Trong định thời (Timing diagram) ở trên, để thực hiện chu kỳ ghi dữ liệu cần: |  | 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 | Trong sơ đồ kết nối hệ thống vi xử lý 8086 trên tín hiệu ALE và các bộ chốt (Latch) được sử dụng để: |  | 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 | Trong sơ đồ kết nối hệ thống vi xử lý hình vẽ trên, tín hiệu IO/M sẽ có mức logic 0: |  | Khi CPU đọc bộ nhớ | Khi CPU truy cập bộ nhớ | Khi CPU ghi bộ nhớ | Khi CPU truy cập vào ra | 2 |
| 140 | Theo bảng trạng thái của bộ nhớ SRAM ở trên, dữ liệu sẽ được lưu vào SRAM khi: |  | 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 |
| 141 | Trong định thời đọc cổng vào ra hình trên, trong chu kỳ đọc vào ra các tín hiệu MEMR và MEMW sẽ ở trạng thái: |  | Logic 0 | Logic 1 | High Z | MEMR mức 0  MEMW mức 1 | 2 |
| 142 | Trong định thời đọc cổng vào ra hình trên, trong chu kỳ ghi vào ra các tín hiệu MEMR và MEMW sẽ ở trạng thái: |  | Logic 0 | Logic 1 | High Z | MEMR mức 0  MEMW mức 1 | 2 |
| 143 | Giản đồ thời gian hình vẽ trên là giản đồ của chu kỳ: |  | Đọc bộ nhớ | Ghi bộ nhớ | Đọc vào ra | Ghi vào ra | 2 |
| 144 | Trong định thời hình trên, trong chu kỳ ghi vào ra các tín hiệu IOR và IOW sẽ ở trạng thái: |  | Logic 0 | Logic 1 | High Z | MEMR mức 0  MEMW mức 1 | 2 |
| 145 | Trong định thời hình trên, trong chu kỳ ghi vào ra các tín hiệu IOR và IOW sẽ ở trạng thái: |  | Logic 0 | Logic 1 | High Z | MEMR mức 0  MEMW mức 1 | 2 |
| 146 | Trong định thời ở trên, dữ liệu được ghi vào bộ nhớ EPROM khi cấp xung lập trình vào: |  | Ngõ vào E | Ngõ vào P | Ngõ vào G | Ngõ vào Vpp | 2 |
| 147 | Trong định thời ở trên, tín hiệu G là: |  | 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 |
| 148 | Trong sơ đồ kết nối IO hình vẽ trên, vi mạch 16L8 thực hiện chức năng: |  | Giải mã địa chỉ | Chốt địa chỉ | Truyền dữ liệu | Cung cấp tín hiệu điều khiển vào ra | 2 |
| 149 | Thiết bị điện tử hình trên có chức năng: |  | 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 | Trong sơ đồ trên, MOSFET có chức năng: |  | 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 | Mạch giao tiếp relay hình vẽ trên có chức năng: |  | 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 | Trong sơ đồ hình vẽ trên diode D1 có chức năng: |  | 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 | Trong sơ đồ trên, PORT1 sẽ có địa chỉ: |  | A0H | A1H | A4 | A2 | 3 |
| 154 | Trong sơ đồ trên, PORT1 sẽ có địa chỉ: |  | A0H | A1H | A4H | A3H | 3 |
| 155 | Trong sơ đồ trên để truy cập tới các cổng vào ra cho phép từ bộ giải mã địa chỉ A7A6 phải có giá trị là: |  | 00B | 01B | 10B | 11B | 3 |
| 156 | Trong sơ đồ hình vẽ trên bộ nhớ 2716 có vùng địa chỉ: |  | FF800H-FFFFFH | FF000H – FF7FFH | FF700H – FFFFFH | FF900H- FFFFFH | 3 |
| 157 | Trong sơ đồ trên BUS dữ liệu của CPU là: |  | 8 bit | 16 bit | 32 bit | 64 bit | 3 |
| 158 | Trong sơ đồ kết nối bộ nhớ hình vẽ trên bộ nhớ #2 có vùng địa chỉ: |  | 00000H-3FFFFH | C0000H-FFFFFH | 80000H-BFFFFH | 40000H-7FFFFH | 3 |
| 159 | Trong sơ đồ kết nối bộ nhớ hình vẽ trên bộ nhớ #3 có vùng địa chỉ: |  | 00000H-3FFFFH | C0000H-FFFFFH | 80000H-BFFFFH | 40000H-7FFFFH | 3 |
| 160 | Trong sơ đồ kết nối bộ nhớ hình vẽ trên sẽ có bao nhiêu bộ nhớ 8KB? |  | 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 | Trường 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 | Trường 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 xuống của các chân GPIO | 1 |
| 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\_DR;  I2C\_SR1;  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 thự ký tự thông báo: trạng thái hệ thống, chế độ hoạt động, … | Hiển thị các băn bản 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 | GPIOC->ODR=(1<<7)| (1<<8)|(1<<9)  Dòng lệnh lập trình thanh ghi ODR của STM32 ở trên thực hiện: |  | 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 | GPIOC->BSRR =(1<<23)| (8<<24)|(8<<25)  Dòng lệnh lập trình thanh ghi BSRR của STM32 ở trên thực hiện: |  | Để đưa chân 7, 8, 9 PORTC lên mức cao | Để đưa chân 23, 24, 25 PORTC lên mức cao | Để đưa chân 23, 24, 25 PORTC về mức thấp | Để đưa chân 7, 8, 9 PORTC về mức thấp | 2 |
| 194 | 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? |  | 100Ω | 150Ω | 200Ω | 250Ω | 2 |
| 195 | Trong hình kết nối nút nhấn vào STM32 hình vẽ trên, điện trở kéo lên (Pull up) có chức năng gì? |  | 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 | 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? |  | 01101001B | 10010110B | 11001001B | 01100110B | 2 |
| 197 | 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? |  | 01101001B | 10010110B | 11001001B | 01100110B | 2 |
| 198 | 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? |  | 1 | 2 | 3 | 4 | 2 |
| 199 | Trong giản đồ thời gian của Text LCD hình trên, dữ liệu sẽ được ghi vào LCD ở: |  | 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 | 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: |  | R4R3R2R1=  0000B | R4R3R2R1=  1111B | R4R3R2R1=  0001B | R4R3R2R1=  1000B | 2 |
| 201 | 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: |  | R4R3R2R1=  0000B | R4R3R2R1=  1111B | R4R3R2R1=  0001B | R4R3R2R1=  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\_GetValue; HAL\_ADC\_PollForConversion;  HAL\_ADC\_Stop | HAL\_ADC\_Start;  HAL\_ADC\_Init();  HAL\_ADC\_ConfigChannel;  HAL\_ADC\_PollForConversion;  HAL\_ADC\_GetValue;  HAL\_ADC\_Stop | HAL\_ADC\_Init();  HAL\_ADC\_ConfigChannel;  HAL\_ADC\_Start;  HAL\_ADC\_PollForConversion;  HAL\_ADC\_GetValue;  HAL\_ADC\_Stop | HAL\_ADC\_Init();  HAL\_ADC\_ConfigChannel;  HAL\_ADC\_PollForConversion;  HAL\_ADC\_Start;  HAL\_ADC\_GetValue;  HAL\_ADC\_Stop | 2 |
| 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\_GetStateHAL\_SPI\_GetError | 2 |
| 208 | Trong sơ đồ kết nối, dữ liệu từ Master cung cấp tới chân MOSI sẽ: |  | 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 tăng trên SDA; Chế độ nhanh; Gửi bit ACK | Tốc độ, chu kỳ nhiệm vụ của clock; Chế độ địa chỉ; Địa chỉ Slaver; Chống nhiễu; Gọi riêng; Thời gian tăng trên SDA; Chế độ nhanh; Gửi bit ACK | Tốc độ, chu kỳ nhiệm vụ của clock; Chế độ địa chỉ; Địa chỉ Slaver; Chống nhiễu; Gọi tổng quát; Thời gian tăng trên SDA; Chế độ nhanh; Gửi bit ACK | Tốc độ, pha của clock; Chế độ địa chỉ; Địa chỉ Slaver; Chống nhiễu; Gọi tổng quát; Thời gian tăng trên SDA; Chế độ nhanh; Gửi bit ACK | 2 |
| 213 | Lệnh I2C1->OAR1 = (0x50 << 1) viết cho STM32 sẽ khởi động: |  | Khối I2C1 là Slaver có địa chỉ là 80 | Khối I2C1 là Slaver có địa chỉ là 50 | Khối I2C1 là Master có địa chỉ là 80 | Khối I2C1 là Master có địa chỉ là 50 | 2 |
| 214 | Các hàm trong thư viện HAL cho cổng 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 | Trong sơ đồ kết nối trên, dữ liệu từ Master cung cấp tới chân SDA sẽ: |  | 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\_INT;  HAL\_UART\_Receive\_INT;  HAL\_UART\_Transmit\_DMA;  HAL\_UART\_Receive\_DMA;  HAL\_UART\_IRQHandler; | 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\_IRQHandler; | 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;  HAL\_UART\_IRQHandler; | 2 |
| 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 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 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 gián đoạn 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 | #include "Device/Include/stm32f10x.h"  void gpio\_init(void)  {  RCC->APB2ENR= (1<<3)  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);  } }  Chương trình viết dưới lớp thanh ghi của STM32 thực hiện: |  | 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 |
| 226 | #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;  for(i=5;i<=12;i++)  {  GPIOB->ODR= (1<<i); delay(500);  GPIOB->ODR&= ~(1<<i);  delay(500);  } }  Chương trình viết dưới lớp thanh ghi của STM32 thực hiện: |  | Điều khiển các LED nối Kathode vào cổng B tắt dần | Điều khiển các LED nối Anode vào cổng B tắt dần | Điều khiển các LED nối Anode vào cổng B sáng dần | Điều khiển các LED nối Kathode vào cổng B sáng dần | 3 |
| 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 | #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;  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\_PORT, BUTTON\_PIN);  if (buttonState == GPIO\_PIN\_RESET) { HAL\_GPIO\_TogglePin(LED\_PORT, LED\_PIN);  HAL\_Delay(100);  }}}  Chương trình viết dưới lớp thư viện HAL cho STM32 thực hiện: |  | Điến khiển LED sáng khi nhấn nút | Chuyển trạng thái sáng tắt của LED khi nút nhấn tác động | Điến khiển LED sáng khi không nhấn nút | Điều khiển LED chớp tắt khi nhấn nút | 3 |
| 229 | #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\_PORT, DIGIT\_1\_PIN DIGIT\_2\_PIN|DIGIT\_3\_PIN|DIGIT\_4\_PIN, GPIO\_PIN\_RESET);  switch (digit) {  case 1: HAL\_GPIO\_WritePin(DIGIT\_PORT, 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++) { 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);  } }}}  Chương trình viết dưới lớp thư viện HAL cho STM32 thực hiện: |  | 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 |
| 230 | 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);  Hàm LCD\_W cho STM32 dưới lớp thư viện HAL ở trên thực hiện: |  | 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 | 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++) {  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);  } }  Sau khi gọi hàm quét bàn phím ma trận viết dưới lớp thư viện HAL cho STM32 ở trên sẽ được: |  | 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; 0010B; 0100B; 1000B | 3 |
| 232 | 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(); }  Đoạn chương trình STM32 dưới lớp thư viện HAL trên thực hiện: |  | 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 | 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 | 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 | 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 | 3 |
| 233 | #include <stdio.h>  #define NUM\_ROWS 4  #define NUM\_COLS 4  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);  }  Sau đoạn chương trình trên sẽ được biến “key” bằng |  | 4 | 1 | Ký tự A | 0x0A | 3 |
| 234 | #include <stdio.h>  int main() {  int num = 1642;  char str[10];  snprintf(str, sizeof(str), "%d", num);  printf("Result: %s\n", str);  return 0;  }  Sau khi thực hiện chương trình C trên biến str có giá trị là: |  | 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 10mV/oC, điện áp tại 0oC là 0.5V. Giá trị độc được là 123 sẽ tương ứng với nhiệt độ là: |  | 24.748 °C | 34.748 °C | 44.748 °C | 54.748 °C | 3 |
| 236 | #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\_IRQn, 0, 0);  HAL\_NVIC\_EnableIRQ(TIM2\_IRQn);  HAL\_TIM\_Base\_Start\_IT(&htim);  }  HAL\_TIM\_PeriodElapsedCallback(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();  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) {  } }  Chương trình STM32 dưới lớp thư viện HAL ở trên thực hiện: |  | 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 |
| 237 | #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(&htim, &sConfigOC, TIM\_CHANNEL\_1);  HAL\_TIM\_PWM\_Start(&htim, TIM\_CHANNEL\_1);  }  int main(void) {  HAL\_Init();  Timer\_PWM\_Config();  while (1) {  }}  Chương trình STM32 dưới lớp thư viện HAL ở trên thực hiện: |  | 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 |
| 238 | #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(uint16\_t GPIO\_Pin) {  if (GPIO\_Pin == GPIO\_PIN\_0) {  }}  int main(void) {  HAL\_Init();  EXTI\_Config();  while (1) {  }}  Chương trình STM32 dưới lớp thư viện HAL ở trên chương trình ngắt sẽ thực hiện khi: |  | 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 |
| 239 | 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);  }  Hàm STM32 dưới lớp thư viện HAL ở trên thực hiện: |  | Khởi động Timer2 hoạt động ở chế độ encoder | Khởi động Timer2 hoạt động ở chế độ đếm sự kiện ở kênh số 1 | Khởi động Timer2 hoạt động ở chế độ one -pulse tạo ra xung ở kênh PWM1 | Khởi động Timer2 hoạt động ở chế độ tạo ra xung PWM ở kênh PWM1 | 3 |
| 240 | #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;  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\_DIV1; HAL\_TIM\_Encoder\_Init(&htim2);  }  static void MX\_GPIO\_Init(void) {  }  Chương trình STM32 dưới lớp thư viện HAL ở trên thực hiện: |  | 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 | Cấu hình Timer2 ở chế độ Encoder;  Khởi động đọc giá trị Encoder ở kênh 1;  Lấy giá trị Encoder ở chương trình chính | Cấu hình Timer2 ở chế độ Encoder;  Khởi động đọc giá trị Encoder ở kênh 2;  Lấy giá trị Encoder ở chương trình chính | Cả ba câu kia đề đúng | 3 |
| 241 | Các lệnh nào sau đây của ARM thực hiện việc rẽ nhánh chương trình? |  | B  BL | BX  BLX | BXJ | Cả ba câu kia đề 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 dời 10 bit | Dịch trái logic thanh ghi độ dời 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 dời 10 bit | Dịch trái logic thanh ghi độ dời 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 dời đi 10 bit | Dịch phải logic thanh ghi độ dời 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 dời 10 bit | Dịch phải thanh ghi độ dời 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 dời tỷ lệ (Register with scaled register offset), địa chỉ dữ liệu cần truy cập là: |  | Tổng thanh ghi cơ sở và thanh ghi độ dời đã được cộng thêm một giá trị | Tổng thanh ghi cơ sở và thanh ghi độ dời đã được chia tỷ lệ | Tổng thanh ghi cơ sở và thanh ghi độ dời đã được trừ đi một giá trị | Tổng giữa thanh ghi cơ sở với thanh ghi độ dời 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 << 2) | r1 = (r1 + r2) << 2  r0 = [r1] | r1 = r1 + (r2 << 2)  r0 = [r1] | r0 = [r1]  r1 = (r1 + r2) << 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 << 2) | r1 = (r1 + r2) << 2  r0 = [r1] | r1 = r1 + (r2 << 2)  r0 = [r1] | r0 = [r1]  r1 = (r1 + r2) << 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 rẻ 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 | Sau khi thực hiện lệnh:  MOV R0, #42  CMP R0, #10  Lệnh rẻ nhánh nào sẽ thực hiện rẽ nhánh chương trình: |  | BEQ | BGT | BVS | BLT | 2 |
| 302 | foo:  ADD R0, R1, R2 ;  BX LR ;  MAIN:  MOV R1, #5 ;  MOV R2, #3 ;  BL foo  Trong đoạn chương trình trên sau khi thực hiện lệnh “BX LR” điều khiển chương trình sẽ: |  | Chuyển về sau lệnh “BL foo” | Chuyển về nhãn MAIN | Chuyển về nhãn foo | Chuyển tới lệnh “BL LR” | 2 |
| 303 | .thumb  foo:  ADD R1, R1, #5 ;  BX LR ;  .code 32  MAIN:  MOV R2,#10;  MOV R1, #12 ;  BLX foo ;  MOV R2,R1  Sau khi thực hiện đoạn chương trình trên R2 có giá trị bằng bao nhiêu? |  | #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]  Đoạn chương trình hợp ngữ ở trên tương đương khi viết trong C: |  | x=a+b+c | x=a-b-c | x=a+b-c | x=-a+b-c | 3 |
| 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  Các lệnh hợp ngữ trên tương đương với cấu trúc lập trình nào trong C? |  | if | if … else | switch …case | while | 3 |
| 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]  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: |  | 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 |
| 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 | 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  Chương trình hợp ngữ ARM trên tương đương với cấu trúc lập trình nào trong C? |  | while | do … while | for | switch …case | 3 |
| 315 | .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  MOV R7, #1  SWI 0  Chương trình hợp ngữ ARM trên tương đương với cấu trúc lập trình nào trong C? |  | while | do … while | for | switch …case | 3 |
| 316 | .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  CMP R2, R1  BLT outer\_loop  MOV R7, #1  SWI 0  Chương trình hợp ngữ ARM trên thực hiện công việc: |  | 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 |
| 317 | .text  .global main  main:  LDR R0, =TCode  BXJ R0  B end  TCode:  @ đoạn code  end:  MOV R7, #1  SWI 0  Trong chương trình ARM ở trên, đoạn lệnh sau nhãn “Tcode” là: |  | 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 | .data  Input .word 42  Result .word 0  Buffer .space 10  .text  .global main  main:  LDR R0, =Input  LDR R1, =Buffer  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  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: |  | 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 trở lại thành chuỗi ký tự hệ 16 và lưu trong biến “Result” | 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 lại thành số nguyên và lưu trong biến  “Buffer” | 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 nguyên và lưu trong biến “Result” | 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 trở lại thành chuỗi ký tự hệ 16 và lưu trong biến “Buffer” | 3 |
| 319 | .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  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  Chương trình hợp ngữ trên thực hiện công việc: |  | Tìm giá trị nhỏ nhất lưu vào M2 giá trị lớn nhất lưu vào M1 | Tìm giá trị nhỏ nhất lưu vào M1 giá trị lớn nhất lưu vào M2 | Tìm giá trị lớn nhất lưu vào M2 | Tìm giá trị nhỏ nhất lưu vào M1 | 3 |
| 320 | 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]  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  Chương trình hợp ngữ trên thực hiện công việc: |  | Cấp Clock cho GPIOD;  Khởi động GPIOD chân 12 là Input;  Nhận dữ liệu từ GPIOD chân 12 | Cấp Clock cho GPIOD;  Khởi động GPIOD chân 12 là Output;  Bật tắt LED gắn trên GPIOD chân 12 | Cấp Clock cho GPIOD;  Khởi động GPIOD chân 12 là Output;  Bật LED gắn trên GPIOD chân 12 | Cấp Clock cho GPIOD;  Khởi động GPIOD chân 12 là Output;  Cấp mức logic 0 ra GPIOD chân 12 | 3 |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | **GIẢNG VIÊN BIÊN SOẠN**  *(Ký, ghi rõ họ tên)*  **Phạm Thế Duy** |