확인 문제 정답

01-1 컴퓨터 구조를 알아야 하는 이유

1. ②

컴퓨터 구조를 이해하는 것은 프로그램을 빠르게 구현하는 것과는 큰 관련이 없습니다.

2. 미지의 대상, 분석의 대상

컴퓨터 구조를 이해하면 컴퓨터를 미지의 대상에서 분석의 대상으로 인식하게 됩니다.

01-2 컴퓨터 구조의 큰 그림

1. 데이터, 명령어

프로그램을 이루는 정보에는 데이터와 명령어가 있습니다.

2. ③

CPU는 프로그램 속 명령어를 해석하고 실행하는 장치입니다.

3. 메모리

프로그램이 수행되려면 반드시 메모리에 저장되어 있어야 합니다.

4.

① 보조기억장치	㉡ 보관할 프로그램 저장
② 메모리	㉠ 실행되는 프로그램 저장

메모리는 실행되는 프로그램을 저장하는 용도로, 보조기억장치는 전원이 꺼져도 저장한 내용을 기억하기에 보관할 프로그램을 저장하는 용도로 사용됩니다.

5. ①

02-1 0과 1로 숫자를 표현하는 방법

1. 2

1GB는 1000MB이므로, 2000MB는 2GB와 같습니다.

2. ④

1byte는 1024bit가 아니라 8bit와 같습니다.

3. $0011_{(2)}$

2의 보수는 모든 0과 1을 뒤집고, 여기에 1을 더함으로써 얻을 수 있습니다.

$1101_{(2)}$의 모든 0과 1을 뒤집으면 $0010_{(2)}$, 여기에 1을 더하면 $0011_{(2)}$이므로 $1101_{(2)}$의 2의 보수는 $0011_{(2)}$입니다.

4. $11011010_{(2)}$

십육진수를 이진수로 변환할 때 십육진수 한 글자를 네 개의 이진수로 간주하면 간편합니다.

십육진수 D를 이진수로 표현하면 $1101_{(2)}$, 십육진수 A를 이진수로 표현하면 $1010_{(2)}$이므로,

십육진수 0xDA를 이진수로 표현하면 이를 이어 붙인 $11011010_{(2)}$이 됩니다.

5. ①

이진수를 많이 사용하는 이유 중 하나는 십육진수 간의 변환이 쉽기 때문입니다.

02-2 0과 1로 문자를 표현하는 방법

1. hongong

아스키 코드 표에 따르면 h는 103, o는 111, n은 110, g는 103으로 인코딩됩니다.

2. ②

EUC-KR은 완성형 인코딩입니다.

3. 1110 1100 1001 0101 1000 1000 1110 1011 1000 0101 1001 $0101_{(2)}$

'안'과 '녕'은 각각 3바이트로 인코딩됩니다. '안'에 부여된 값 1100 0101 0100 $1000_{(2)}$을 UTF-8 3바이트 인코딩 형식에 넣으면 1110 1100 1001 0101 1000 $1000_{(2)}$, '녕'에 부여된 값 1011 0001 0101 $0101_{(2)}$을 UTF-8 형식에 넣으면 1110 1011 1000 0101 1001 $0101_{(2)}$입니다.

따라서 이를 이어 붙인 값이 '안녕'을 UTF-8로 인코딩한 값입니다.

03-1 소스 코드와 명령어

1. ③, ④

기계어와 어셈블리어는 고급 언어가 아닌 저급 언어입니다.

2. ②

인터프리터 언어는 한 줄 한 줄 저급 언어로 해석하며 실행해야 하기 때문에 일반적으로 컴파일 언어보다 느립니다.

03-2 명령어의 구조

1. ②

연산 코드 필드에는 명령어가 수행할 내용이 담깁니다. 오퍼랜드 필드에는 메모리의 주소가 담깁니다.

2. ① 6, ② 200

메모리 6번지 속 200이라는 값을 CPU로 가지고 옵니다.

레지스터 간접 주소 지정 방식은 연산 코드에 사용될 데이터를 메모리에 저장하고, 그 주소(유효 주소)를 저장한 레지스터를 명령어 오퍼랜드 필드에 명시하는 방법입니다. 즉, R1이라는 레지스터에 담긴 메모리 주소를 확인하면 연산 코드에 사용될 데이터를 얻을 수 있습니다.

R1 레지스터에 담긴 주소는 6번지로, 200을 저장하고 있습니다. 따라서 위 명령어를 레지스터 간접 주소 지정 방식으로 실행하면 메모리 6번지의 200이라는 값이 CPU로 읽어 들여집니다.

04-1 ALU와 제어장치

1. −3

부호 플래그가 음수입니다. 이 말은 곧 계산 결과가 음수임을 나타냅니다. 계산 결괏값, 즉 $101_{(2)}$은 $011_{(2)}$에 2의 보수를 취하여 음수로 표현한 값이고, $011_{(2)}$은 십진수 3이므로 계산 결과를 십진수로 표현하면 −3이 됩니다.

2. ④

CPU 내에서 연산을 담당하는 부품은 ALU입니다.

3. ① 플래그 레지스터, ② 명령어 레지스터, ③ 제어 버스

제어장치는 플래그 레지스터로부터 플래그 값을 받아들이고, 명령어 레지스터로부터 현재 해석할 명령어를 받아들입니다. 그리고 제어 버스를 통해 제어 신호를 내보냅니다.

4. ④

CPU의 구성 요소에는 크게 레지스터, ALU, 제어장치가 있습니다. 하드 디스크는 CPU의 구성 요소가 아닙니다.

04-2 레지스터

1. 2100번지

프로그램 카운터는 다음으로 실행할 명령어의 주소를 저장합니다.

2. ① 플래그 레지스터, ② 프로그램 카운터, ③ 범용 레지스터, ④ 명령어 레지스터

3. 6번지

스택 포인터는 스택에 남아 있는 최상단의 데이터 주소를 저장합니다. 그림 속 스택에서 데이터를 두 번 빼내면 6번지에 저장된 3밖에 남지 않습니다. 따라서 스택 포인터는 6번지를 저장합니다.

4. ④

레지스터는 CPU 내에 있는 작은 임시 저장 장치입니다.

04-3 명령어 사이클과 인터럽트

1. ④

인터럽트는 프로그램의 순차적인 실행을 끊습니다.

2. ③

인터럽트 비트로 모든 인터럽트를 막을 수 있는 것은 아닙니다. 막을 수 없는 인터럽트를 NMI라고 합니다.

3. 인터럽트 서비스 루틴

CPU는 인터럽트를 처리하기 위해 인터럽트 서비스 루틴을 실행합니다. 하던 작업을 잠시 백업하고 인터럽트 서비스 루틴을 실행한 뒤 다시 실행을 재개합니다.

4. ②

메모리에 있는 명령어를 CPU로 가져오는 단계를 인출 사이클(fetch cycle)이라고 합니다.

5. ③

하드웨어 인터럽트가 발생하면 CPU는 수행하던 작업을 잠시 백업한 뒤 인터럽트 서비스 루틴을 실행합니다. 인터럽트 서비스 루틴의 실행이 끝나면 백업해 둔 작업을 복구하여 다시 수행을 재개합니다.

05-1 빠른 CPU를 위한 설계 기법

1. ③

클럭 속도를 지나치게 높이면 발열 문제가 발생합니다.

2. ④

멀티스레드 프로세서는 하나의 코어로도 여러 개의 하드웨어 스레드를 처리할 수 있습니다.

3. ① 하드웨어, ② 소프트웨어

스레드에는 하드웨어적 스레드와 소프트웨어적 스레드가 있습니다.

4. 코어

코어는 CPU 내에서 명령어를 처리하는 부품이고, CPU는 이 코어를 여러 개 가질 수 있습니다. 이와 같은 CPU를 멀티코어 CPU라고 부릅니다.

05-2 명령어 병렬 처리 기법

1. ②

슈퍼스칼라는 여러 개의 명령어 파이프라인을 이용하는 기법입니다.

2. ① 비순차적 명령어 처리, ② 슈퍼스칼라

비순차적 명령어 처리 기법은 순서를 바꾸어 명령어를 실행하는 기법이고, 슈퍼스칼라 기법은 여러 개의 파이프라인을 이용하여 명령어를 동시에 처리하는 기법입니다.

05-3 CISC와 RISC

1. ④

CISC는 다양한 명령어 주소 지정 방식을 지원합니다.

2. ①

RISC는 CISC보다 많은 명령어로 프로그램을 실행합니다.

06-1 RAM의 특징과 종류

1. ③

메모리로 사용되는 저장 장치는 SRAM이 아니라 DRAM입니다. SRAM은 일반적으로 '대용량으로 만들어질 필요는 없지만 속도가 빨라야 하는 저장 장치', 가령 캐시 메모리에서 사용됩니다.

2. ① DDR SDRAM, ② SRAM, ③ DRAM, ④ SDRAM

대역폭을 두 배 넓힌 SDRAM은 DDR SDRAM입니다. 그리고 시간이 지나도 저장된 데이터가 사라지지 않는 RAM은 SRAM, 데이터의 소멸을 막기 위해 일정 주기로 데이터를 재활성화해야 하는 RAM은 DRAM, 클럭과 동기화된 DRAM은 SDRAM입니다.

3. ① SRAM, ② DRAM, ③ DRAM, ④ SRAM

SRAM은 주로 캐시 메모리로 활용되고 집적도가 DRAM에 비해 낮습니다. DRAM은 주로 주기억장치로 활용되며, 대용량화하기 유리합니다.

4. ① 2, ② 2

DDR3 SDRAM은 DDR2 SDRAM에 비해 대역폭이 두 배 넓은 RAM이고, DDR2 SDRAM은 DDR SDRAM에 비해 대역폭이 두 배 넓은 RAM입니다.

06-2 메모리의 주소 공간

1. ③

프로그램이 실행될 때마다 다른 주소에 적재될 수 있습니다. 매 실행마다 동일한 주소에 적재되는 것은 아닙니다.

2. ③

실행 중인 프로그램 각각에 부여된 0번지부터 시작되는 주소는 물리 주소가 아닌 논리 주소입니다.

3. ①

MMU는 논리 주소를 물리 주소로 변환해 주는 장치입니다.

4. ① 한계 레지스터, ② 베이스 레지스터

한계 레지스터는 명령어가 다른 프로그램 범위를 침범하는지 검사합니다. 베이스 레지스터는 논리 주소와 더해져 물리 주소로 변환되는 데에 사용됩니다.

06-3 캐시 메모리

1. ① 레지스터, ② 캐시 메모리, ③ 메모리, ④ 보조저장장치

본문의 그림을 참고해 보세요.

2. ④

캐시 히트율이 높으면 캐시 메모리의 활용도가 높아 성능이 좋아집니다.

07-1 다양한 보조기억장치

1. ① 플래터, ② 스핀들, ③ 헤드

하드 디스크에서 데이터가 저장되는 요소는 플래터입니다. 그리고 그 플래터는 스핀들이 돌리고, 이를 읽고 쓰는 구성 요소는 헤드입니다.

2. ②

TLC 타입은 MLC 타입보다 읽고 쓰는 속도가 느립니다.

07-2 RAID의 정의와 종류

1. ②

보조기억장치에는 수명이 있습니다.

2. ③

RAID 0은 데이터를 단순히 병렬적으로 스트라이핑하여 분산 저장하는 방식입니다.

3. RAID 6

본문의 그림을 참고해 보세요.

확인 문제 정답

08-1 장치 컨트롤러와 장치 드라이버

1. ①, ③

장치 컨트롤러를 사용하는 첫 번째 이유는 입출력장치 종류가 많아 규격화가 어렵기 때문이고, 두 번째 이유는 입출력장치는 일반적으로 CPU, 메모리보다 속도가 느리기 때문입니다.

2. ②

전원이 꺼져도 대용량의 데이터를 저장하는 것과 장치 컨트롤러는 관계가 없습니다.

3. ① 장치 컨트롤러 , ② 프로그램

장치 드라이버는 장치 컨트롤러가 컴퓨터 내부와 정보를 주고받을 수 있도록 하는 프로그램입니다.

4. ②

컴퓨터가 장치 드라이버를 인식하고 실행할 수 있다면 해당 입출력장치의 사용이 가능합니다.

08-2 다양한 입출력 방법

1. ②

고립형 입출력은 입출력장치를 위한 주소 공간을 별도의 공간으로 간주하기 때문에 메모리의 주소 공간이 축소되지 않습니다.

2. A, B

인터럽트 A의 인터럽트 서비스 루틴을 실행하던 도중 우선순위가 더 높은 인터럽트 B가 발생했을 때 CPU는 인터럽트 A의 인터럽트 서비스 루틴을 잠시 멈추고, 인터럽트 B의 인터럽트 서비스 루틴을 실행합니다.

3. ③

DMA 컨트롤러는 CPU를 거치지 않고 메모리와 입출력장치 간의 데이터를 주고 받습니다.

4. 메모리 맵 입출력

메모리 맵 입출력은 메모리에 접근하기 위한 주소 공간과 입출력장치에 접근하기 위한 주소 공간을 하나의 주소 공간으로 간주하는 입출력 방식입니다.

5. 고립형 입출력

고립형 입출력은 메모리에 접근하기 위한 주소 공간과 입출력 장치에 접근하기 위한 주소 공간을 별도의 주소 공간으로 분리하는 입출력 방식입니다.

09-1 운영체제를 알아야 하는 이유

1. 자원

운영체제는 실행할 프로그램에게 자원을 할당하고, 프로그램이 올바르게 실행되도록 돕는 프로그램입니다.

2. ②

운영체제는 커널 영역에 적재됩니다.

09-2 운영체제의 큰 그림

1. 커널

커널은 운영체제의 핵심 기능을 담당하는 부분을 의미합니다.

2. ④

시스템 호출은 일종의 인터럽트입니다.

3. ③

사용자 인터페이스 제공은 운영체제의 핵심 서비스라 보기 어렵습니다.

10-1 프로세스 개요

1. ④

사용자와 상호작용하지 않는 프로세스도 있습니다.

2. ①

문맥 교환이 지나치게 빠르게 반복되면 문맥 교환에 드는 오버헤드로 인해 좋지 않습니다.

10-2 프로세스 상태와 계층 구조

1. ① 생성, ② 준비, ③ 실행, ④ 종료, ⑤ 대기

본문의 그림을 참고해 보세요.

2. ① 로그인 프로세스, ② Vim 프로세스, ③ bash 프로세스

bash는 로그인 프로세스의 자식 프로세스이고, Vim은 bash 프로세스의 자식 프로세스입니다.

3. ①

fork 시스템 호출과 폴더 생성은 관련이 없습니다.

4. ④

준비 상태는 CPU를 할당받기를 기다리고 있는 상태입니다.

10-3 스레드

1. ②

프로세스 내의 스레드들은 각기 다른 코드/데이터/힙 영역을 가지고 있지 않습니다.

2. 공유하지 않지만, 공유합니다

프로세스끼리는 기본적으로 자원을 공유하지 않지만, 프로세스 내의 스레드끼리는 같은 프로세스 내의 자원을 공유합니다.

11-1 CPU 스케줄링 개요

1. ④

선점형 스케줄링은 프로세스가 이용 중인 자원을 빼앗을 수 있는 스케줄링 방식입니다.

2. ① 준비 큐, ② 대기 큐

본문의 그림을 참고해 보세요.

3. ②

선점형 스케줄링은 문맥 교환 과정의 오버헤드가 비선점형 스케줄링에 비해 더 큽니다.

11-2 CPU 스케줄링 알고리즘

1. ③

선입 선처리 스케줄링 알고리즘은 준비 큐에 삽입된 순서대로 CPU를 할당합니다.

2. ① 기아 현상, ② 에이징 기법

우선순위가 낮아 실행이 계속 연기되는 문제를 기아 현상이라고 하며, 이를 예방하기 위한 기법이 에이징 기법입니다.

12-1 동기화란

1. ① 실행 순서 제어, ② 상호 배제

프로세스를 올바른 순서대로 실행하는 동기화를 실행 순서 제어를 위한 동기화라고 하며, 동시에 접근해서는 안 되는 자원에 하나의 프로세스만 접근하게 하는 동기화를 상호배제를 위한 동기화라고 합니다.

2. ①

임계 구역을 여러 개의 프로세스가 동시에 실행하면 레이스 컨디션이 발생합니다.

12-2 동기화 기법

1. ④

반드시 바쁜 대기를 할 필요는 없고, 대기 상태로 접어들게 할 수도 있습니다.

2. 스레드 B

스레드 B가 조건 변수 y에 대해 대기 상태(wait)에 있으므로 y.signal을 호출하면 스레드 B가 실행됩니다.

3. 상호 배제, 실행 순서 제어

세마포를 이용하면 동시에 실행되는 프로세스 혹은 스레드 간에 상호 배제를 위한 동기화와 실행 순서 제어를 위한 동기화를 할 수 있습니다.

13-1 교착 상태란

1. ④

식사하는 철학자 문제에서 모든 철학자가 동시에 식사를 하려고 하면 교착 상태가 발생합니다.

2. 상호 배제, 점유와 대기, 비선점, 원형 대기

교착 상태가 발생할 네 가지 조건은 상호 배제, 점유와 대기, 비선점, 원형 대기입니다.

3. ②

자원 할당 그래프가 원의 형태를 띄고 있기 때문입니다.

확인 문제 정답

13-2 교착 상태 해결 방법

1. ②

교착 상태가 발생해도 이를 회복할 수 있습니다.

2. 5개

P2에 자원 두 개를 할당하면 남은 자원은 3-2=1개가 되며, P2가 실행을 성공적으로 종료하여 자원을 반납하면 1+4=5개가 됩니다.

3. ②

타조 알고리즘은 교착 상태가 발생해도 이를 무시하는 방법입니다.

4. ④

원형으로 대기하면 교착 상태가 발생할 수 있습니다.

14-1 연속 메모리 할당

1. ① 최초 적합, ② 최악 적합, ③ 최적 적합

최초 적합은 최초로 발견한 적재 가능한 빈 공간에 프로세스를 배치하는 방식, 최악 적합은 프로세스가 적재될 수 있는 가장 큰 공간에 프로세스를 배치하는 방식, 최적 적합은 프로세스가 적재될 수 있는 가장 작은 공간에 프로세스를 배치하는 방식입니다.

2. ④

외부 단편화가 발생한 공간에는 프로세스를 배치할 수 없습니다.

3. ④

메모리 스와핑은 메모리에서 사용되지 않는 일부 프로세스를 보조기억장치로 내보내고 실행할 프로세스를 메모리에 적재하는 방식입니다.

4. ①

연속 메모리 할당은 외부 단편화를 야기합니다.

14-1 페이징을 통한 가상 메모리 관리

1. ④

TLB 히트의 경우 메모리 접근을 한 번으로 줄일 수 있습니다.

2. ③

2번 페이지의 유효 비트가 1이므로 해당 페이지는 현재 메모리에 적재되어 있습니다.

3. ②

페이지 테이블은 페이징 기법을 사용하는 컴퓨터에서 사용합니다.

4. ①

TLB는 페이지 테이블의 캐시 메모리입니다.

14-3 페이지 교체와 프레임 할당

1. 3회

본문의 그림을 확인해 보세요.

2. ④

페이지 폴트율 기반 프레임 할당은 페이지 폴트율의 상한선보다 적게 프레임을 할당하는 방식입니다.

15-1 파일과 디렉터리

1. ④

확장자는 파일의 유형을 파악하기 위한 정보입니다.

2. ① minchul/c.tar, ② /home/minchul/c.tar

절대 경로는 루트 디렉터리부터 시작하는 경로이고, 상대 경로는 현재 디렉터리부터 시작하는 경로입니다.

3. ①

디렉터리도 파일과 동일하게 보조기억장치에 저장되어 있지 않습니다.

확인 문제 정답

15-2 파일 시스템

1. ②

보조기억장치 내에 파일을 연속적인 블록으로 할당하는 방식은 연결 할당입니다.

2. ④

FAT 파일 시스템은 다음 블록 주소를 FAT에 명시합니다.

3. ①

유닉스 파일 시스템은 색인 할당 기반의 파일 시스템입니다.

4. ①

파티셔닝과 포매팅 작업을 거쳐야 파일 시스템을 이용할 수 있습니다.