C basic language

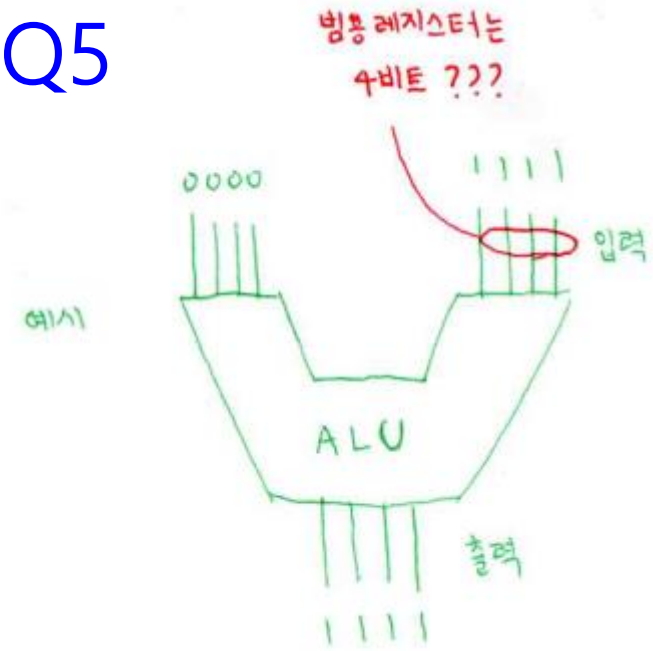
임베디드스쿨 2기

Lv1과정

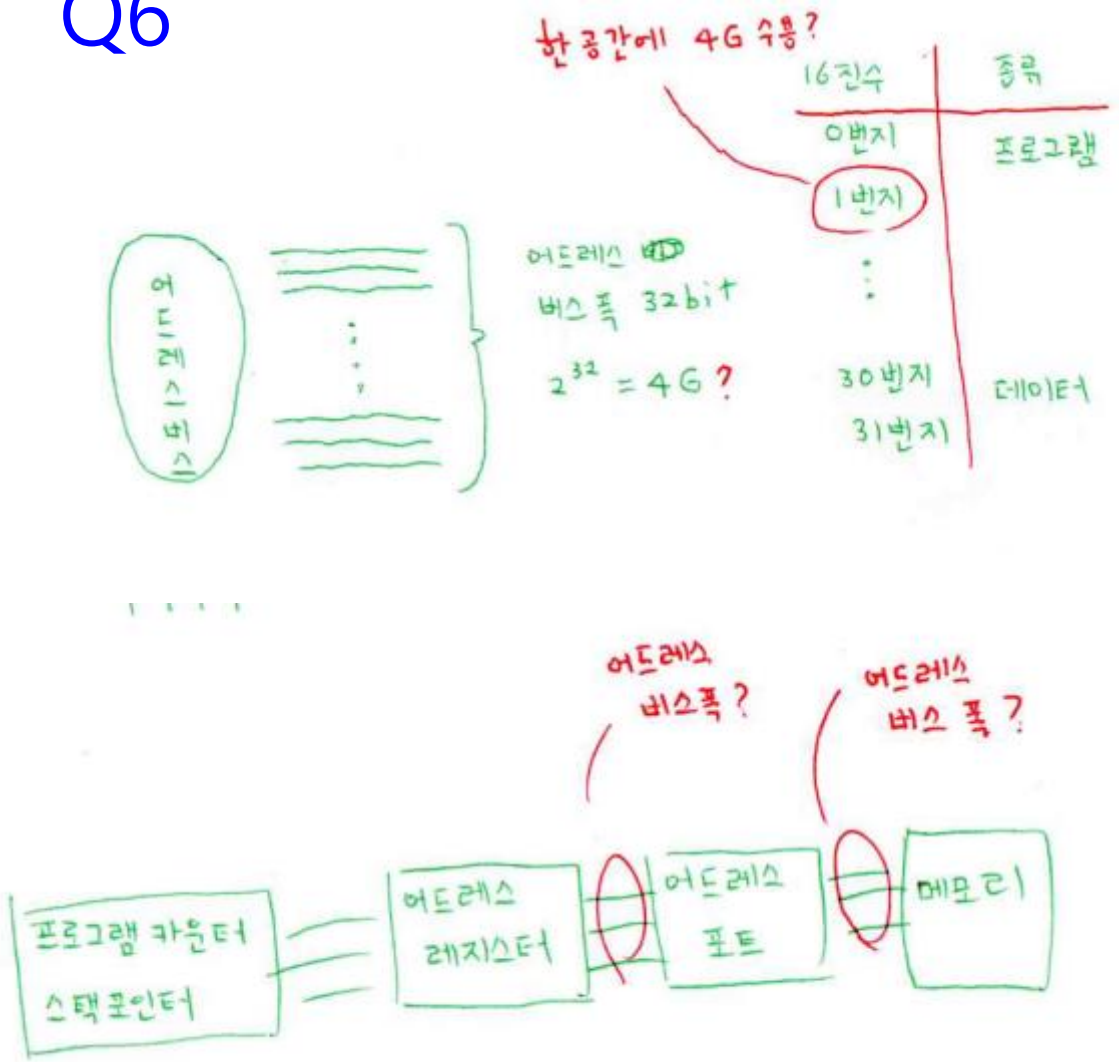
2021. 03. 26

김효창

Q5


범용 레지스터는
4비트 ???
0000
1111
입력
예시
ALU
출력
1111

Q6


한 공간에 4G 수용?
16진수
종류
0번지
프로그램
1번지
어드레스 버스
어드레스 버스폭
버스폭 32bit
2^{32} = 4G ?
30번지
31번지
데이터
어드레스 버스폭 ?
어드레스 버스 폭 ?
프로그램 카운터
스택포인터
어드레스 레지스터
어드레스 포트
메모리

Question

Q1. 메모리 공간 2바이트를 할당 받아서 0x1234 를 쓰게 되면 Little Endian 방식을 사용한다고 표기되어 있습니다

물리적으로 데이터를 조작하거나 산술 연산을 수행할 때에는 Little Endian 을 사용

데이터의 각 바이트를 배열처럼 취급할 때, 데이터를 전송할 때, 소켓 통신을 할 때 Big Endian 을 사용

- 데이터를 조작한다는 의미가 예를 들어 전역 변수, 정적변수, 문자열을 사용하였을 때인가요?

(code, data, heap, stack 에서 data 영역에 해당되는 모든 것들)

- 산술 연산은 Little Endian 방식을 적용할 때 + , - , * , / , % 의 연산자들만 해당될까요? (피연산자는 제외??)
- 배열에는 Big Endian 을 사용한다고 되어 있습니다 (예를 들어 배열 변수 여러 개를 메모리 공간에 바이트 단위로 할당하였을 때 Big Endian 방식이 사용되는 건가요?
- 사용자인 프로그래머가 Little Endian/ Big Endian 를 코드 영역에서 정의할 수 있나요?

(효율적으로 자동으로 지정되기 때문에 엔디안 방식은 무시해도 되는 부분 인가요?)

Q2. 함수 포인터 변수 : 특정한 프로토타입의 함수 주소를 저장할 수 있는 메모리 공간

예시로 int main (void) 빨간색 부분이 프로토타입의 함수인가요???

위에 int main 이 프로토타입의 함수라면 이것은 가상 메모리인 stack 영역에 할당 되는 것 일까요??

Q3. C 프로그램에서 보는 메모리는 가짜다 !

우리는 프로그래밍을 하면 DRAM을 활용하는 것으로 알고 있지만 실제는 가상 메모리를 활용하였다 라는 의미인가요?

가상메모리 : RAM , 하드디스크 등을 동시에 합쳐서 하나의 구조로 보이도록 해주는 메모리

Code, Data, Stack, Heap

Question

Q4. 어드레스 레지스터가 32 bit 가 있다 지정할 수 있는 번지 수는 4G 까지 이다 하드웨어 8G 연결하면 메모리 낭비이다 ,
어드레스 레지스터는 MAR 이라고 불리는 CPU 레지스터라고 명시되어 있습니다 위에서 하드웨어라는 것은
어드레스 레지스터 IC (SN54173) 인가요?? (어드레스 포트와 데이터 포트에 연결된 외부 메모리??)

Q5. 수강 전에는 운영체제 32 bit / 64 bit 를 시스템 속성에서 확인하였습니다
위에 있는 bit 도 ALU 범용 레지스터에 의해 결정되는 것일까요? (사진 첨부)

Q6. 어드레스 버스 폭으로 어드레스를 몇 비트 취급할 수 있는지 알 수 있다고 되어 있었습니다
어드레스 버스 폭 의미가 전체적인 것을 얘기하는 것인지 한 공간 안에 Bit 를 얘기하는 것인지 잘 모르겠습니다
어드레스 버스 폭이라는 것이 Q4 어드레스 레지스터에서 파생된 것인가요??
(사진 첨부)

Q7. DRAM 의 동작 원리에서 word line 은 메모리 셀 사용 여부를 결정하고 bit line 은 메모리 값을 확인하고 저장될 값을
정해주는 역할이라고 되어 있었습니다. DRAM 뿐만 아니라 여러 종류 메모리와 기타 등등 에서 이러한 방식을 자주
사용할까요???
이러한 과정을 명령 읽기(페치) , CPU의 명령 처리 동작 중에 하나일까요? (페치 , 디코드 , 실행)

Question

Q8. 가상 메모리의 크기는 왜 포인터의 크기를 따라가는가?

가상 메모리를 64비트로 표현하면 8바이트를 모두 표현할 수 있었던 것입니다

결과는 범용 레지스터 64 bit 를 알맞게 활용하지 못해서 손해 아닌가요?? (64bit 모두를 활용해야 손해 X ?)

가상 메모리가 어떤 까닭에 효율적인 것인지 말씀해 주셨는데 잊어버렸습니다....

Q9. 포인터가 아무 것도 가리키고 있지 않을 때는 int *p = NULL; 로 설정

포인터를 사용하기 전에 NULL 인지 아닌지를 확인

포인터를 사용하지 않는다면 삭제하면 될텐데 저러한 방법을 사용하는 이유가 있나요?

NULL 확인 여부는 터미널 명령어를 찾아서 디버그 모드에서 확인하는 건가요?? (gdb??)

Q10. ALU 범용 레지스터 64 bit 에서는 포인터의 크기가 8 Byte

int *p = &data 의 *p 라는 변수는 다른 변수의 주소를 가지고 있으면서 앞에 자료형 int 4바이트 무시하고

8바이트의 크기를 가지고 있다 라고 이해하면 될까요?