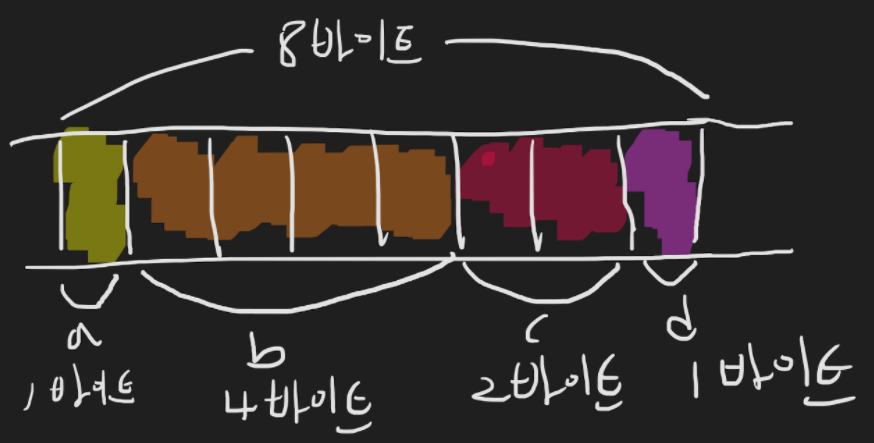
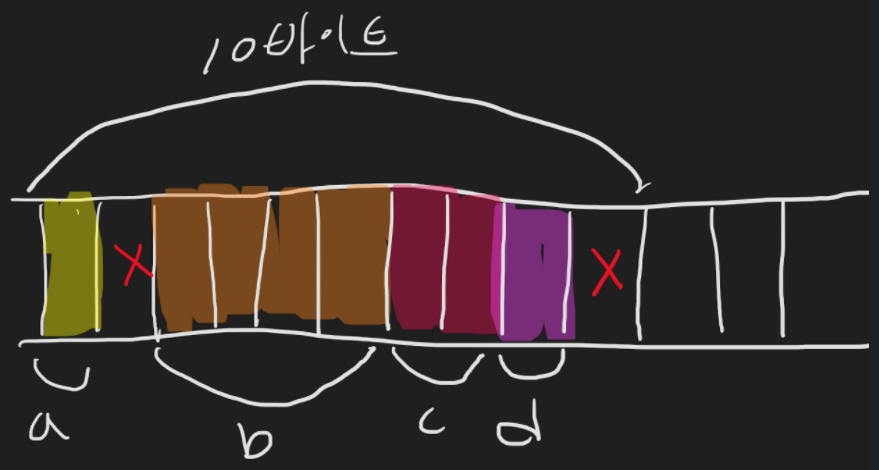
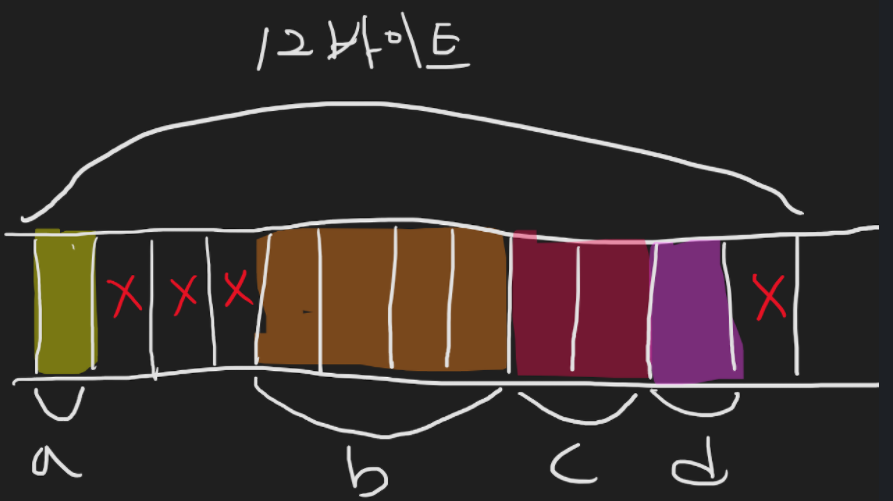
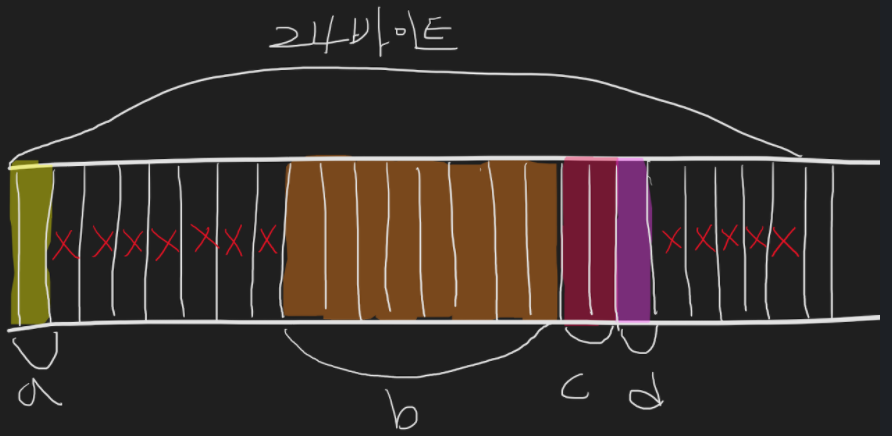
**Struct Member Alignment**

1. **구조체 멤버 정렬**- 컴파일러에서 기본값으로 구조체 정렬 기준을 어떻게 설정하는지에 따라 구조체로 만든 자료형의 크기가 달라진다.  
   텍스트, 칠판이(가) 표시된 사진

   자동 생성된 설명텍스트, 칠판이(가) 표시된 사진

   자동 생성된 설명
2. **1Byte 정렬**- 이 정렬을 사용하면 우리가 설정했던 구조체의 본래 의미대로 메모리가 구성된다.  
   - 즉 Test1의 자료형의 크기는 8Byte고, Test2의 크기는 12Byte이다.
3. **2Byte 정렬**- 2Byte 정렬은 전체 크기가 2의 배수가 되어야 한다.  
   - 1Byte 자료형 요소들은 1Byte씩의 사용하지 않는 공간이 생긴다.  
   - Test1 구조체의 크기는 10Byte
4. **4Byte 정렬**- 4Byte 정렬은 전체 크기가 4의 배수가 되어야 한다.  
   - 자료형의 크기가 4의 배수가 아니라면 1~3Byte를 버리게 된다.  
   - Test1 구조체의 크기는 12Byte
5. **8Byte 정렬**- 8Byte정렬은 전체 크기가 8의 배수가 되어야 한다.  
   - 자료형이 8Byte보다 작은 경우에는 해당 요소의 크기로 정렬된다.  
   - Test2의 경우, a = 1Byte, b = 8Byte이므로, 16Byte에 2개의 요소가 정렬된다.  
   - Test2 구조체의 크기는 총 24Byte가 된다.
6. **결론**- 요즘 컴파일러들은 8Byte 정렬을 기본 값으로 하고 있다.  
   - 구조체에 8Byte 크기의 자료형을 사용하지 않았다면 낭비가 적겠지만,  
   - double이나, \_\_int64 같은 8Byte 자료형을 사용하는 순간 구조체의 크기가 커진다.  
   - 따라서 이러한 구조체 정렬 원리를 파악하여 구조체의 요소는 같은 크기끼리 모아준다.  
   - 즉, 구조체를 선언할 때, 요소의 순서를 같은 크기끼리 배치한다는 것

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명  
- 기존 Test2의 자료형 크기는 24Byte였으나, 순서만 바꾸면 16Byte로 변경된다.  
- 낭비되는 Byte가 12Byte에서 4Byte로 감소한다.