# Find Prime Number

이 문제의 조건은 다음과 같다.

1. 5개의 수가 주어졌을 때, 소수가 몇 개인지 찾아서 출력
2. 직접 5개의 수를 입력
3. 주어진 수들 중 소수의 개수를 출력
4. 음수는 입력하지 않는다.

이 문제에서 고려해야 하는 점은 크게 3가지이다. 1. 두자리 수 이상 정수 입력, 2. 해당 숫자의 소수 여부 판단, 3. 총 소수의 개수 count

여기서도 ascending sort와 마찬가지로 숫자를 저장하기 위한 레지스터를 r7-r11로 설정하였고 코드가 실행될 때 초기화 해주었다. 이에 대한 설명도 ascending sort에서 자세히 한다.

이번 문제를 풀기 위해서 먼저 숫자 5개를 입력 받는다. 이후 하나씩 소수 여부를 파악하고 해당 숫자가 소수라면 r0에 1을 더하고, 소수가 아니라면 r0 값을 유지하여 총 소수 개수를 count 한다.

1. 두 자리 수 입력 코드

텍스트, 영수증, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

음수 입력이 존재하지 않아 오히려 ascending sort에 비해 간단하다. 이에 대한 매커니즘은 ascending sort 1에서 1-2, 1-3을 제외한 1-1과 동일하므로 넘어가도록 하겠다.

이 과정을 총 5개의 입력을 반복하여 r7-r11까지 숫자들을 저장해준다.

1. 소수 여부 판단. & 소수 count

텍스트, 폰트, 화이트, 스크린샷이(가) 표시된 사진

자동 생성된 설명소수를 찾기 위해 r2에 2를 저장하여 환경을 세팅한다. 또한 소수 여부를 파악하고자 하는 숫자가 1이라면 이후 알고리즘에서 적용이 되지 않아 다음 숫자로 넘어간다. 이때 1은 소수가 아니므로 따로 count 가 증가하지 않는다.

소수인지 확인하기 위하여 2부터 1씩 증가시키면서 해당 숫자까지 존재하는 모든 자연수로

나누어 나머지가 0인 지점이 존재하는지 확인하였다. 소수는 1과 자기자신만으로 나누어 떨어지는 2이상의 자연수이기 때문에 숫자를 증가시키며 나눴을 때 중간에 나누어 떨어진다면 해당 숫자는 소수가 아니게 된다.

이를 구현한 코드는 아래와 같다.



텍스트, 영수증, 스크린샷, 대수학이(가) 표시된 사진

자동 생성된 설명

#소수 여부를 확인 하고자 하는 r7에 있는 값을 r1에 저장한다. 이후 div\_mod를 사용하여 r1/r2 작업을 수행해준다. 이때 사용한 나눗셈 코드는 아래와 같다.

텍스트, 영수증, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

나눗셈 자체의 개념은 ascending sort에서 자세히 설명하였고, 이번 코드에서는 나머지가 0인지 아닌지 여부가 중요하기 때문에 나머지의 의미에 대해 파악을 해보겠다.

Start\_div에서 r2<r1일 때 sub r1, r1, r2에서 r1에서 r2를 반복하여 빼고, r2>r1일 때 뺄셈을 종료한다. 이 작업을 거친다면 최종적으로 r1에 저장되는 값은 나눗셈에서 나머지의 의미와 동일하다. 우리는 이 r1에 저장된 값이 0인지 여부로 소수 여부를 확인한다.

다시 소수 확인 여부로 돌아가보자. cmp r1, #0 // beq mod\_num1\_is\_0 코드에서 Div\_mod 에서 가지고 온 r1에 저장된 나머지가 0일 때 mod\_num1\_is0으로 분기하고, 나머지가 0이 아니라면 add r2, r2, #1 // b num1\_loop\_i에 의해 r2에 1을 더해 num1\_loop\_i 를 반복한다. 이 과정이 피제수를 2로 나누고, 3으로 나누고, 4, 5, 6,…,피제수 로 나누는 과정을 가능하게 해준다. 나누면서 언제든 나머지가 0인 경우가 생긴다면 cmp r1, #0 // beq mod\_num1\_is\_0 코드에서 mod\_num1\_is0으로 분기하게 된다. 어찌 됐든 제수가 피제수 본인이 될 때 까지 해당 과정을 반복하면 결국은 나머지가 0이 될 것이므로 최종적으로 mod\_num\_is\_0으로 분기된다. mod\_num\_is\_0로 분기 되었다면 r7에 저장된 피제수와 r2에 저장된 값이 같은지 확인을 한다. 다르다면 바로 다음 숫자로 넘어가여 위 과정과 동일하게 소수 여부를 확인한다.

텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진

자동 생성된 설명같다면 cmp r7, r2 // beq count1을 통해 count1으로 분기하여, r0에 저장된 값에 1을 더해 다시 저장한다.

Count 이후 다음 숫자로 분기하여 다음 숫자의 소수 여부를 확인한다.

이와 같은 과정을 5개 숫자들에 대해 모두 수행을 해준다면 최종적으로 5개 숫자 중 소수가 몇 개 존재하는지 r0에 저장된다.

텍스트, 영수증, 폰트, 대수학이(가) 표시된 사진

자동 생성된 설명많아봤자 5개가 나올 것이기 때문에 출력은 한자리 정수로 충분하고, loop\_end에서 r0에 저장된 값을 print\_char를 이용해 출력하면 console 창에 소수의 개수가 보인다.

이때, 소수의 개수임을 명확하게 하기 위해 console 창에서 [n]으로(n은 소수의 개수) 보이게 작업을 하였다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명위 과정들을 통해 소수의 개수를 찾는 코드를 구현했고,

여러가지 예시들을 수행해본 결과는 아래와 같다.

좌측이 입력, 우측 [ n ] 이 소수의 개수이다.

한자리 수에 대해서도 정상 작동하며

두 자리 수, 0이 혼합되어 있는 수들의 조합에서도

정상 작동함을 알 수 있다.

첫번째, 두번째 예시는 ppt예제 이외에 다른 숫자들을 입력하여 예제 이외에도 잘 돌아가는 지 확인을 해보았다.