미니 대회(2019 년 12 월 17 일)

다음 임무 1 과 임무 2 를 수행하는데 총 실행시간(싸이클)을 최대한 단축하시오.

[임무 1]

1. 설명

data[] 배열 안의 C_START 부터 N_DATA 개의 원소에 대해서 두 번째 자리인 10 의 자리 수(0-9)의 빈도를 3 개 그룹으로 조사하여 count[] 배열에 저장하시오.(C_START 가 0 이면, 배열의 첫 원소 data[0]에서 시작한다.)

- 그룹 A: 1,2,9 → count[0]을 증가
- 그룹 B: 0,4,5,6 → count[1]을 증가
- 그룹 C: 3,7,8 → count[2]를 증가

예: data[i]의 값이 2457 이라면 10 의 자리수가 5 이므로, 그룹 B 인 count[1]을 1 만큼 증가한다.

```
C_START     BOX     0
N_DATA      RESBOX  1

COUNT:
COUNT0      RESBOX  1
COUNT1      RESBOX  1
COUNT2      RESBOX  1
P_DATA      BOX     DATA

            ORG     9000
DATA        BOX     2345, 3427, ...
             . . .
```

2. 실행

“IN IO_BRIDG”로 입력 받은 값 4 개(TERM1, TERM2, TERM3, TERM4)가 있으며 C_START 와 N_DATA 는 아래와 같이 결정된다.

- C_START = 0 // 고정 값이며 채점할 때 바뀔 수 있음
- N_DATA = TERM1+TERM2+TERM3+TERM4

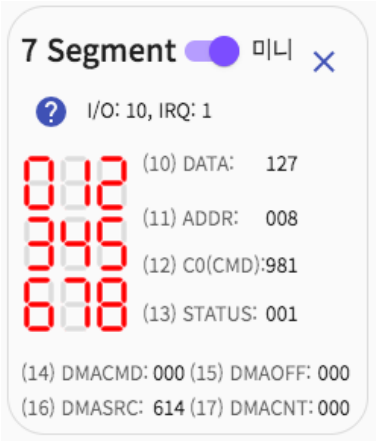
3. 채점

- 각 그룹(A, B, C)의 최종 결과를 IO_BRIDGE 에 출력한다.
- 출력은 count[0], count[1], count[2] 순으로 한다.

```
LD A, COUNT0
OUT IO_BRIDGE
LD A, COUNT1
OUT IO_BRIDGE
LD A, COUNT2
OUT IO BRIDGE
```

[임무 2]

[임무 1] 에 중간 단계를 추가하여 각 단계에서 count[0], count[1], count[2]의 값을 7SEGIO 로 변환하여 아래와 같이 출력하시오.



- 3 x 3 의 7SEGIO 를 생성한다.

```
//$DEV SEVEN-SEGMENT-D 10 3 3
```

- 중간 결과는 다음 주기로 3 회 생성하여 출력한다.
    - 중간 1: C_START=0, NDATA= TERM1
    - 중간 2: C_START=0, NDATA += TERM2
    - 중간 3: C_START=0, NDATA += TERM3
- 각 단계(중간 1, 중간 2, 중간 3)에서 출력은 count[0], count[1], count[2]의 순으로 한다.
- count[0], count[1], count[2]의 값은 7SEGIO 에 아래와 같이 배치되며 각 세그먼트의 값이 이전 값과 같아도 출력한다.(ADDR 은 7SEGIO 의 버퍼 주소)
    - count[0] : ADDR0 ~ ADDR2
    - count[1] : ADDR3 ~ ADDR5
    - count[2] : ADDR6 ~ ADDR8
- 최종 결과는 다음과 같이 생성되며 [임무 1]에서 설명한 것처럼 7SEGIO 가 아니라 IO_BRIDGE 로 출력한다.
    - 최종: C_START=0, NDATA += TERM4

  예를 들어 TERM1, TERM2, TERM3, TERM4 의 값이 50, 40, 60, 50 이면 총 200 개의 데이터에서 빈도를 누적해서 세는데 TERM1 이 끝나면 50 개(data[0]-data[49]), TERM2 가 끝나면 누적해서 90 개(data[0]-data[89]), TERM3 이 끝나면 누적해서 150 개(data[0]-data[149]) 그리고 TERM4 가 끝나면 최종 누적 200 개(data[0]-data[199])의 데이터를 처리한 것이다.

[주의 사항]

- 100~199 번지는 MMIO 영역이니 여기에 어떤 코드나 데이타가 로딩되지 않도록 주의한다.
- 코드 전체 범위가 1000 번지를 넘는 경우 addressing 에 매우 주의해야 하므로 가급적 모든 코드의 범위가 1000 번지 안에 들어올 수 있도록 작성한다. (상황에 따라 ORG 500 등은 제거)
- 코드 전체 범위가 1000 번지를 넘는 경우 리터럴 사용에 주의하거나 가능하면 사용하지 않는다.
- 스택, Heap, 처리하려는 데이터 배열 등은 1000 번지 범위 밖에 자유롭게 배치할 수 있다. 이 경우에도 Direct Absolute Addressing 으로 접근하지 않도록 주의한다. 가능하지만 수업에서 다루지 않았으니 대신 Indirect Addressing 을 사용한다.

[실제 대회 당일 변경될 수 있는 조건]

- 그룹(A,B,C)의 개수 또는 구성[임무 1]
- 빈도를 세는 자리 수[임무 1]
- TERM1, TERM2, TERM3, TERM4 의 값[임무 2]

[평가 방법]

1. 참가 점수: 10 점
    1) 강의실로 와야 함

2. [임무 1] 성공: +20 점
    1) 부분정답으로 표시됨

3. [임무 1] ,[임무 2] 통합: +30 점
    1) 정답으로 표시됨
    2) 순차 실행해도 됨

4. 싸이클 경쟁 순위에 따른 점수
    1) 1 등: 100 점
    2) 2 등: 97 점
    3) 3 등: 94 점

5. 싸이클 경쟁 분포에 따른 점수
    1) 정답자의 평균 싸이클에서 표준편차 구간 보다 빠르면: +40 점(100 점)
    2) 정답자의 평균에서 빠른 쪽으로 표준편차 구간 사이 이면: +30 점(90 점)
    3) 정답자의 평균에서 느린 쪽으로 표준편차 구간 사이 이면: +20 점(80 점)
    4) 정답자의 평균에서 느린 쪽으로 표준편차 1,2 구간 사이: +10 점(70 점)

** 4 번(순위), 5 번(분포) 중 더 높은 점수를 반영