1. How process the MIPS Instructions

- bitmasking을 사용하여 16진수로 입력된 instr을 2진수로 변환하고, 상위 6비트에 대하여 opcode에 대입합니다.
- opcode에 따라서 R-format, J-format, I-format을 구분하고, 각 format에 맞게 rs,rt,rd,shmt,funct의 값을 각각 세팅합니다.
- I-format의 경우 주솟값에 대한 연산이 필요하기 때문에, immed value 부분에 대하여 bin_to_hex 변수를 통해 hex값으로 변환합니다.
- 이후 funct, opcode 값을 통해 어떤 명령인지 구분하고 연산을 수행합니다.
  - Sra 연산의 경우, 음수인 경우에 한해 1로 right shift를 수행하므로, sign_bit 변수를 통해 음수인지, 양수인지 구분합니다. 양수일 경우 0, 음수일 경우 1이 대입됩니다.
    - 1을 shamt_num만큼 right shift하고, 이 값을 mask로 설정한 후 temp와 or 연산을 통해 sra연산을 수행합니다.
  - slt 연산의 경우, 음수에 대해서 크기가 제대로 동작하지 못하는 문제가 있었습니다. 이를 해결하기 위해 registers[rs_num], [rt_num]을 unsigned int -> int로 강제 형변환을 수행하여 연산합니다
  - 마찬가지로 beq, bne연산에서도 음수인 경우 연산이 잘 되지 않는 문제가 있어 immed_num에 대해 int16_t로 형변환을 수행합니다.
  - Lw, sw는 각각 메모리의 주솟값을 레지스터에 쓰거나, 레지스터의 값을 메모리에 써야하므로 이때 주솟값의 연산을 위해 registers[rs_num]+immed_hex를 수행합니다.
    - Lw : 이후 big endian 방식으로 memory의 값을 읽고, word에 memory값을 저장한 후 레지스터에 씁니다.
    - Sw : 이후 big endian 방식으로 word(register에 담긴 값)변수에 있는 값을 memory 변수에 대입하는 방식을 통해 메모리에 씁니다.

2. How load the program into the memory

- 파일을 한줄 씩 읽어, 맨 첫번째 token 값만 hexvalue 변수에 대입한후, hexvalue를 1바이트씩 bieg endian 방식으로 memory에 저장합니다
  - 바이트의 위치는 pc값 및 반복문의 반복 변수 i(1바이트씩 저장하기 위해)로 결정합니다
- 메모리로 어셈블리 명령을 load 한 후 pc 값을 4바이트 증가하여 다음 명령어의 위치를 가르키도록 합니다
- 파일의 모든 내용을 처리한 후, 프로그램이 종료되었다는 의미에서 0xffffffff을 추가합니다.

3. How run the program

- memory[]에 있는 값을 읽어서 instruct 변수에 저장합니다.
- Instruct변수를 0xffffffff(half)를 만날때까지 반복하며 pc값을 증가시키고, instruct를 process_instruction() 함수를 통해 명령어를 MIPS로 해석되도록 합니다

4. What program-hidden does

이중 반복문의 형태로, t1의 값을 *(a0+0)에서 로드하고, t1값을 *(sp+8)에 저장한 후, a0=t1 대입 연산을 a0 != (종료조건) 일때까지 반복하는 것으로 보인다. 이 반복문 안에 a1의 값이 a1—되며 4번 loop를 도는 반복문이 있고, t0의 값이 1씩 증가하는데 만약 v0 ==0 이되면 loop가 중단된다. 이 loop 안에서는 0x109c에 있는 프로그램들이 반복적으로 실행된다. loop가 중단되면 스택에서 t1, a0값을 로드하고 (t1 = *(sp+8), a0 = *(sp+4)) a0+=4 연산을 반복적으로 수행하는 프로그램으로 추정된다.

5. Lessons Learned

- 16진수 & 0xffffffff의 계산결과가 1이면 1, 0이면 0을 추가하는 방식을 통해 16진수를 2진수로 변환할 수 있습니다.

- C언어에서 숫자를 처리하는 과정에서, 16진수+2진수 연산이 불가함을 알았습니다. Shift 연산의 경우 문제가 없지만 특히 더하기 연산에서 16진수+2진수로 연산을 수행하게 되면 계산 결과가 달라집니다. 하나의 진수로 통일을 해서 연산을 수행해야합니다.
- 특정 바이트(비트) 추출을 위해 연산을 수행하는 방법을 배웠습니다. 예를 들면 8비트를 추출하기 위해서는 추출하려는 값을 오른쪽으로 8*i비트(i는 반복 변수)만큼 right shift를 수행하면, 8bit로 hex값을 나눈후 추출 할 수 있습니다.
  - 또한 &0xff를 통해 바이트 단위로 값을 추출하고, 상위 비트에 대해서 모두 0으로 처리되어 부호 비트의 영향을 받지 않도록 만들 수 있습니다.