Question 1: (0 points)
Bank of Questions

Assembly Code for Pointers (V16, V17

The code for function `make_big_endian` in the C programming language is as follows:

```
00  struct page_list {
01                   page_list*       prev;
02                   page_list*       next;
03                   unsigned int     page;
04                  };
SB-05  int  *page_count;
06  char valid[1000];
07
08  void  make_big_endian(page_list **page_pointers)
09  {
10    page_list *page_array;
11    unsigned int      i;
12    ...
13    valid[i-1] = valid[i];
14    *page_pointers++;
15    page_array++;
16    *page_pointers = page_array;
17    *page_count = i;
18     ....
19  }
```

In this code `page_array` is a dinamicaly allocated array of `page_lists` (the actual allocation call is omitted above) and `page_pointers` is an array of pointers to `page_list`. Assume that the variable `page_array` is stored in the stack at the address given by `fp-4`; the global variable `page_count` is at the address given by `gp` and the global array `valid` starts at the address `gp+4`. The parameter passed to `make_big_endian` is the address of the first position of the array of pointers to `page_list` called `page_pointers`. Assume that in this architecture an integer is stored in 32 bits and a memory address also occupies 32 bits. Assume that `i` is in `t0`. Write RISC-V assembly code for each one of the following statements in the program above:

Question 2: (10 points)
`valid[i-1] = valid[i];`

Question 3: (10 points)

```
*page_pointers++;
```

Question 4: (10 points)

```
page_array++;
```

Question 5: (10 points)

```
*page_pointers = page_array;
```

Question 6: (10 points)

```
*page_count = i;
```