1. Compile the code

Please compile the following c code into RV64I (RISC-V) assembly code

```
// Declare
uint32_t  a, b, d;
uint64_t c[0xdeadbeef];

// The code
a = b + c[d + 10];
```

Note:

1. The width of Registers in RV64I is 64 bits
2. c is an array of 64-bit integers.
3. The mappings between variables and registers are: a: s0, b: s1, &c: s2, d: s3

2. Assemble the code

Following the previous question, Please refer to **The document** to assemble your assembly code into machine code (in hex).

3. 2's compelement

The **man page of abs()** said that "Trying to take the absolute value of the most negative integer is not defined". Please explain why we cannot find the absolute value of the most negative integer?

4. Endian

```c
#include <stdio.h>
#define ENDIAN_CHECK \
({ \
    int _a = 1; \
    char *_p = (char *)&_a; \
    *_p; \
})

#define S1 ???
#define S2 ???


int main(void)
{
    if (ENDIAN_CHECK)
        printf(S1 "\n");
    else
        printf(S2 "\n");
    return 0;
}
```

Please fill in the blanks for S1 and S2.

5. Structure

```c
#include <stdint.h>

struct Vec2
{
    uint32_t x;
    uint32_t y;
};

struct Character
{
    uint32_t hp;
    uint32_t mp;
    struct Vec2 position;
};

void move_character(struct Character *c, struct Vec2 movement)
{
    c->position.x += movement.x;
    c->position.y += movement.y;
}
```

`c8763` is a new standard of c programming language. To achieve high performance, there is no `struct` in `c8763` standard. Please rewrite the above makes it acceptable for `c8763` standard. Hint:

1. **Link to the code**
2. Please use compiler `RISC-V (32-bits) gcc 13.2.0`
3. Please examine the assembly code generated by the compiler.

---

6. Function object

```c
#include <stdio.h>

int add(int a, int b){
    return a + b;
}

int main(void)
{
    int *prog = (void*)add;
    for (int i=0; i<20/sizeof(int); i++)
        printf("%08x\n", prog[i]);
    return 0;
}
```

Please provide the response output which generates by the given code when compiled using the x86-64 gcc 13.2 compiler and executed on an x86-64 CPU. Hint:

1. **Link to the code**
2. You can answer the question without understanding any x86-64 assembly code
3. In the link, the hex above each instruction is the corresponding machine code