CS203P Lab 2

January 30, 2024

Question 1.

Write a C program to find out if your machine is Big-Endian or Little-Endian.

Question 2.

Write a C program to find out if your machine is Big-Endian or Little-Endian using the Union data structure.

Question 3.

Write a C program to convert a Big-Endian to Little-Endian and vice-versa. Clearly print the value stored along with its address.

Question 4.

Download the attached *assembly.cpp*, and compile using the command mentioned in the first line of the file. What does the output mean to you?

Question 5.

```
static inline uint32_t convert(uint32_t x)
{
  return (x >> 24) | (x >> 8 & 0xff00) | (x << 8 & /*___1___*/) | (x << 24);
}

static inline uint32_t to_bigendian(uint32_t n)
{
  union {
    int i;
    char c;
  } u = {1};
  return u.c ? /*___2___*/ : /*___3___*/;
}

static inline uint32_t to_littleendian(uint32_t n)
{
  union {
    int i;
    char c;
  } u = {1};
  return u.c ? /*___4___*/ : /*___5___*/;
}
```

1. There are 5 blanks in the above code (represented by /*___1___*/, /*___2___*/, etc.). Fill them up with the appropriate code.
2. What is the purpose of this code?
3. How does this code work?

Question 6.

```c
#include <stdlib.h>
void f(void)
{
  int* x = (int *) malloc(10 * sizeof(int));
  x[10] = 0;
}

int main(void)
{
  f();
  return 0;
}
```

1. Are there any bugs in the above code? If so, what are they?
2. Run the program using valgrind.

   ```
   valgrind -leak-check=yes ./above_program
   ```

3. What does the output mean to you?
4. All lines of the output valgrind produces start with something like ==33836==. What is the significance of this number? HINT: Add a sleep statement in the code and run valgrind again. Now when the process is sleeping, run

   ```
   ps -ef | grep 33836
   ```

   in another terminal. What do you see? (Note that everytime you run, this number will change).

Question 7.

- Read about ripes https://github.com/mortbopet/Ripes. It should be installed on your machine. If not, please install it.
- Ripes documentation: https://github.com/mortbopet/Ripes/blob/master/docs/README.md

- Click on the processor icon on the top left bar (not the larger icon at the left) and select RISC-V, 32-bit, Single-cycle processor.
- Click on the Editor icon at the left bar and write the assembly code

  ```
  addi x1, x0, 10
  ```

- Check the contents of the registers. What do you see?
- What do you see if you reset the simulator (F3), and write the assembly code

  ```
  addi ra, zero, 10
  ```

- Try writing more instructions. For example:

  ```
  f = (g+h) - (i+j);
  ```

Question 8.

Not for evaluation.
Explore: `https://dogbolt.org`
Upload an executable file (a.out) and analyze the output.