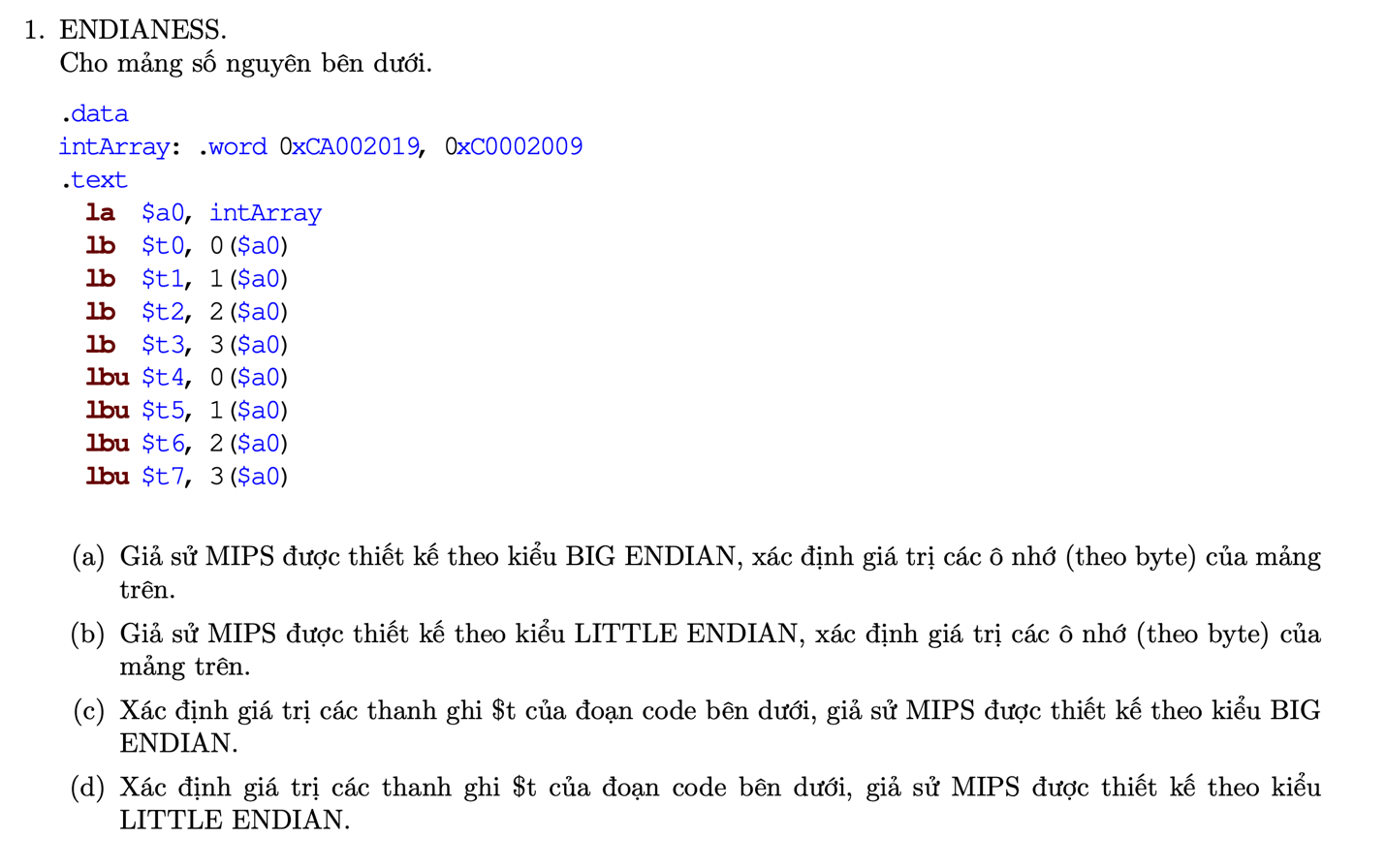
**BÀI LÀM THÊM 1:**



**BÀI LÀM**

(a) BIG ENDIAN

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte 0** | **Byte 1** | **Byte 2** | **Byte 3** |
| 0xCA | 0x00 | 0x20 | 0x19 |
| 0xC0 | 0x00 | 0x20 | 0x09 |

(b) LITTLE ENDIAN

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte 0** | **Byte 1** | **Byte 2** | **Byte 3** |
| 0x19 | 0x20 | 0x00 | 0xCA |
| 0x09 | 0x20 | 0x00 | 0xC0 |

(c) BIG ENDIAN

|  |  |
| --- | --- |
| **Thanh ghi** | **Giá trị** |
| $t0 | 0xFFFFFFCA |
| $t1 | 0x00000000 |
| $t2 | 0x00000020 |
| $t3 | 0x00000019 |
| $t4 | 0x000000CA |
| $t5 | 0x00000000 |
| $t6 | 0x00000020 |
| $t7 | 0x00000019 |

(d) LITTLE ENDIAN

|  |  |
| --- | --- |
| Thanh ghi | Giá trị |
| $t0 | 0x00000019 |
| $t1 | 0x00000020 |
| $t2 | 0x00000000 |
| $t3 | 0xFFFFFFCA |
| $t4 | 0x00000019 |
| $t5 | 0x00000020 |
| $t6 | 0x00000000 |
| $t7 | 0x000000CA |

**BÀI LÀM THÊM 2:**

(a) Theo kiểu BIG ENDIAN, các byte trong một từ 32 bit được lưu trữ theo thứ tự từ cao đến thấp. Do đó, nội dung của vùng nhớ dữ liệu trong chương trình như sau:

int\_1: 0xCA002018 (32 bits)

char\_1: 0xFF (8 bits)

int\_2: 2018 (32 bits)

char\_2: 0xCAFEED (24 bits)

Các lệnh có thể gây ra lỗi khi thực thi là:

- lw $t1, 1($a0): load một .word từ một địa chỉ chưa được căn chỉnh (theo byte) vì địa chỉ bắt đầu không phù hợp để load một .word. Điều này sẽ gây ra lỗi.

- Lệnh lh $t3, 3($a0): load .halfword từ một địa chỉ bắt đầu không phù hợp. Điều này sẽ gây ra lỗi.

Ngoài lề: lệnh lb $t5, 1($a0) không gây ra lỗi, nhưng nó sẽ chỉ tải byte thấp hơn của .word int\_1.

b) Để cải thiện hiệu năng truy cập bộ nhớ, chúng ta có thể sắp xếp lại dữ liệu như sau:

int\_1: .word 0xCA002018

int\_2: .word 2018

char\_1: .byte 0xFF

char\_2: .byte 0xCA, 0xFE, 0xED

Với cách sắp xếp này, tất cả các dữ liệu đều được lưu trữ ở các địa chỉ bắt đầu hợp lệ. Ngoài ra, các dữ liệu cùng loại cũng được lưu trữ gần nhau, giúp cho bộ xử lý có thể truy cập đến nhanh hơn.