MMU

3200102888 米博宇

参数

- 存储器： 32位， 1个
- 大小端模式： 大端
- 寻址方式：8位

代码与测试

```
#include<iostream>
#include<string.h>
/**
 * MMU
 * BigEndian
 * 32bits 1 Memory
 * 8bits adress
 */
using namespace std;
typedef unsigned char byte;
long long MEMSIZE = 1 << 20;
bool BigEndian  = true;
bool LittleEndian= !BigEndian;
class MemoryManagerUnit
{
        public:
          int MemSize;
          int*  Memory;

          bool  Endian;

        MemoryManagerUnit(int size, bool e){
                MemSize = size;
                Memory = new int[size/sizeof(int)];
                memset(Memory, 0, size/sizeof(int));///short[size/sizeof(short)]/by
                Endian = e;
        }

        int lw(int adr){
          int dat0 = Memory[adr / 4];
          int dat1 = Memory[adr / 4 + 1];
          switch(adr & 3){
            case 0: return dat0;
            case 1: return (dat0 << 8) | ((dat1 >> 23) & 255);
            case 2: return (dat0 << 16) | ((dat1 >> 16) & 0xFFFF);
            case 3: return (dat0 << 24) | ((dat1 >> 8) & 0xFFFFFF);
```

```
    }
  }

  void sw(int adr, int dat){
    switch(adr & 3){
    case 0: Memory[adr / 4] = dat; break;
    case 1: Memory[adr / 4] = (Memory[adr / 4] >> 24 << 24) | (dat >> 8);
            Memory[adr / 4 + 1] = (Memory[adr / 4 + 1] & 0xFFFFFF) | (dat <<
            break;
    case 2: Memory[adr / 4] = (Memory[adr / 4] >> 16 << 16) | (dat >> 16);
            Memory[adr / 4 + 1] = (Memory[adr / 4 + 1] & 0xFFFF) | (dat << 1
            break;
    case 3: Memory[adr / 4] = (Memory[adr / 4] >> 8 << 8) | (dat >> 24);
            Memory[adr / 4 + 1] = (Memory[adr / 4 + 1] & 255) | (dat << 8);
            break;
    }
  }

  int lh(int adr){
    int dat0;
    dat0 = Memory[adr / 4];
    switch(adr & 3){
    case 0: return dat0 >> 16;
    case 1: return dat0 << 8 >> 16;
    case 2: return dat0 << 16 >> 16;
    case 3: int dat1 = Memory[adr / 4 + 1];
      return (dat0 << 24 >> 16) | ((dat1 >> 23) & 255);
    }
  }

  int lhu(int adr){
    int dat0;
    dat0 = Memory[adr / 4];
    switch(adr & 3){
    case 0: return (dat0 >> 16) & 0xFFFF;
    case 1: return (dat0 >> 8) & 0xFFFF;
    case 2: return dat0 & 0xFFFF;
    case 3: int dat1 = Memory[adr / 4 + 1];
            return ((dat0 & 255) << 8) | ((dat1 >> 24) & 255);
    }
  }

  void sh(int adr, int dat){
    switch(adr & 3){
    case 0: Memory[adr / 4] = (Memory[adr / 4] & 255) | (dat << 16); break;
    case 1: Memory[adr / 4] = (Memory[adr / 4] & 0xF00F) | (dat << 8); break
    case 2: Memory[adr / 4] = (Memory[adr / 4] & 0xFF00) | dat; break;
    case 3:
            Memory[adr / 4] = (Memory[adr / 4] & 0xFFF0) | (dat >> 8);
            Memory[adr / 4 + 1] = (Memory[adr / 4 + 1] & 0xFFFFFF) | ((dat &
            break;
    }
  }

  int lh(int adr){
```

```
        int lb(int adr){
          switch(adr & 3){
          case 0: return (Memory[adr / 4] >> 24);
          case 1: return ((Memory[adr / 4] << 8) >> 24);
          case 2: return ((Memory[adr / 4] << 16) >> 24);
          case 3: return ((Memory[adr / 4] << 24) >> 24);
          }
          return 0;
        }
        int lbu(int adr){
          switch(adr & 3){
          case 0: return (Memory[adr / 4] >> 24);
          case 1: return ((Memory[adr / 4] << 8) >> 24);
          case 2: return ((Memory[adr / 4] << 16) >> 24);
          case 3: return ((Memory[adr / 4] << 24) >> 24);
          }
        }
        void sb(int adr, int dat){
          switch(adr & 3){
          case 0: Memory[adr / 4] = (Memory[adr / 4] & 0xFFFFFF) | (dat << 24); br
          case 1: Memory[adr / 4] = (Memory[adr / 4] & 0xFF00FFFF) | (dat << 16); b
          case 2: Memory[adr / 4] = (Memory[adr / 4] & 0xFFFF00FF) | (dat << 8); br
          case 3: Memory[adr / 4] = (Memory[adr / 4] & 0xFFFFFF00) | dat; break;
          }
        }

        void init(){
          for(int i = 0; i < MemSize; i++)
            sb(i, i);
        }

        void disp(int adr, int n){
          for(int i=adr; i<n; i++)
            printf("%02X: %02X\n", i, lb(i));
                printf("\n");
        }
};
int main(){
        MemoryManagerUnit* mmu = new MemoryManagerUnit(MEMSIZE, BigEndian);
        cout << "test sb and lb:" << endl;
        mmu->init();
        mmu->disp(0, 16);
//        cout << "------------------------------" << endl;
        cout << "test sw and lw:" << endl;
        mmu->sw(3, 0x12345678);
        mmu->disp(0, 16);
        printf("00: %08X\n04: %08X", mmu->lw(0), mmu->lw(4));
        printf("\n");
        cout << "test sh and lh and lhu:" << endl;
        mmu->init();
        mmu->sh(3, 0x1234);
        mmu->disp(0, 16);
        printf("03: %04X\n", mmu->lh(3));
        printf("03: %04X\n", mmu->lhu(3));
        printf("test is over!");
}
```

```
}
```

测试结果

```
test sb and lb:
00: 00
01: 01
02: 02
03: 03
04: 04
05: 05
06: 06
07: 07
08: 08
09: 09
0A: 0A
0B: 0B
0C: 0C
0D: 0D
0E: 0E
0F: 0F

test sw and lw:
00: 00
01: 01
02: 02
03: 12
04: 34
05: 56
06: 78
07: 07
08: 08
09: 09
0A: 0A
0B: 0B
0C: 0C
0D: 0D
0E: 0E
0F: 0F

00: 00010212
04: 34567807
test sh and lh and lhu:
00: 00
```

```
01: 00
02: 02
03: 12
04: 34
05: 05
06: 06
07: 07
08: 08
09: 09
0A: 0A
0B: 0B
0C: 0C
0D: 0D
0E: 0E
0F: 0F

03: 1268
03: 1234
test is over!
```

结果正确。