计算机系统基础
Programming Assignment

PA 1-2 – 整数的表示和运算

2020年9月17日 / 2020年9月18日

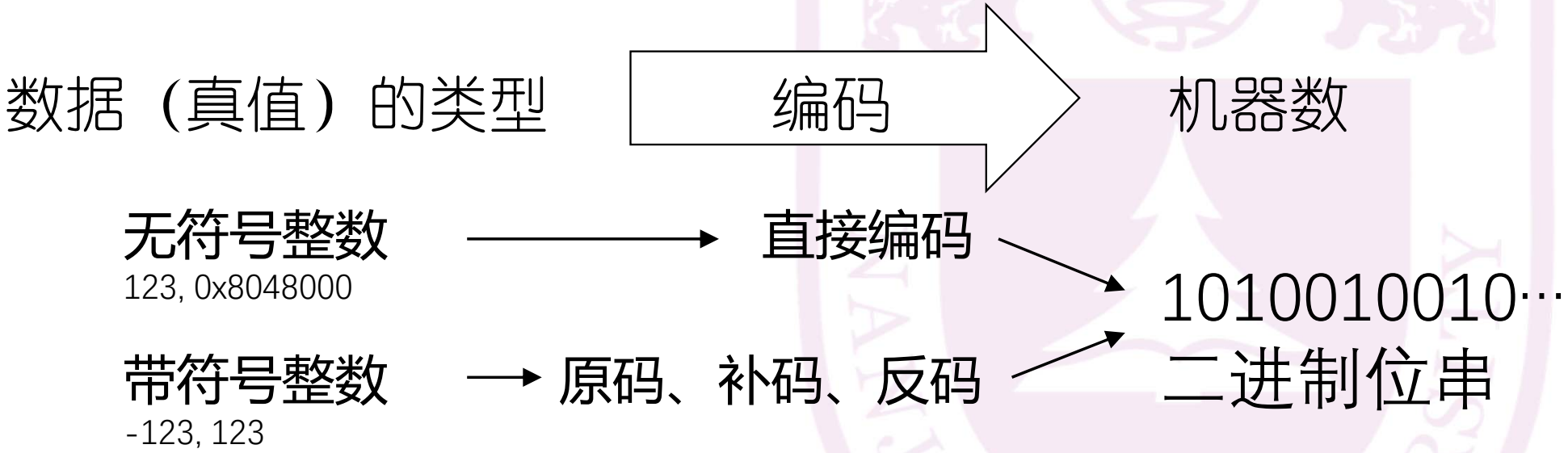
南京大学《计算机系统基础》课程组

目录

整数的表示和运算

- 无符号整数
    - 32位整数：0x0 ~ 0xFFFFFFFF（32个1）
- 带符号整数
    - 原码表示法：最高位为符号位
    - 补码表示法（普遍采用）：各位取反末位加一
        - 用加法来实现减法
        - 可以试试X+(-X)等于多少，其中X为某一32位正整数，-X为其补码表示，运算结果截取低32位

数据的类型及其机器级表示

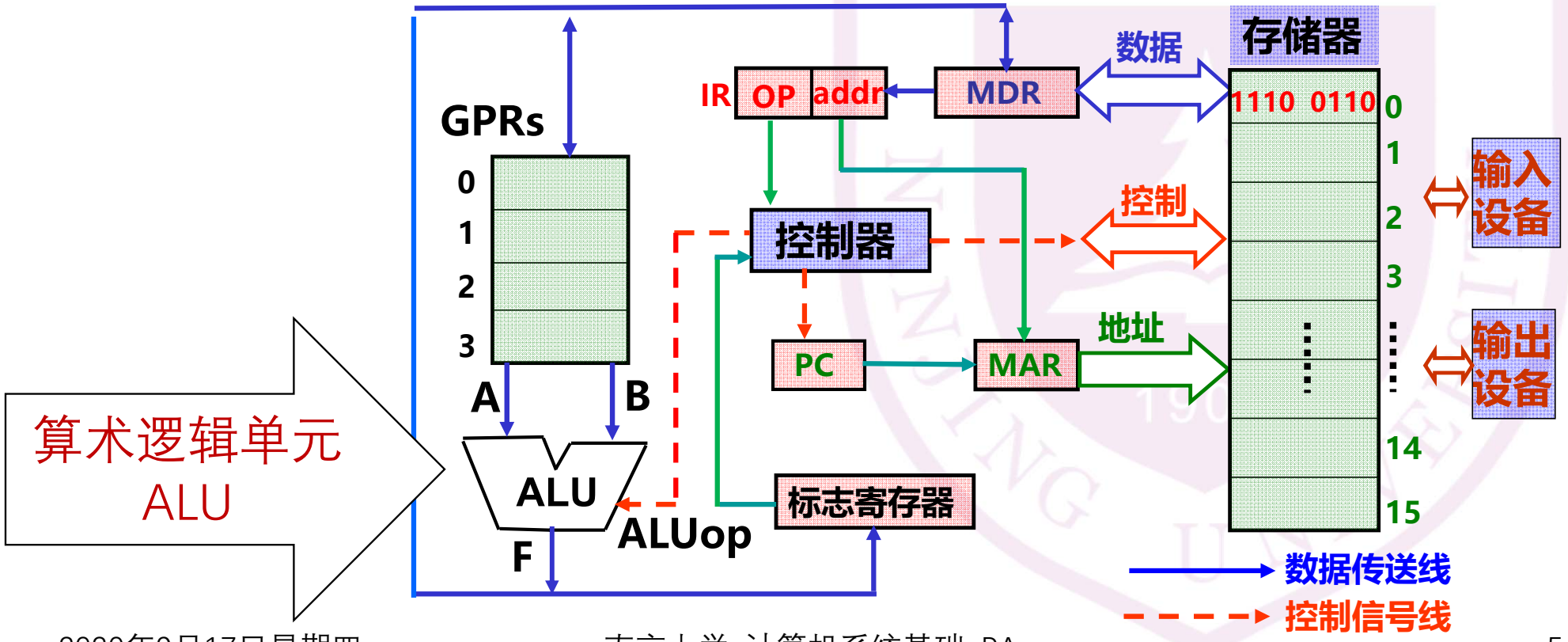


整数的运算部件

CPU：中央处理器；PC：程序计数器；MAR：存储器地址寄存器

ALU：算术逻辑部件；IR：指令寄存器；MDR：存储器数据寄存器

GPRs：通用寄存器组（由若干通用寄存器组成）



算术逻辑单元

- 我们对ALU的功能进行了抽象和包装
  - 能进行各类算术运算：加减乘除、移位
  - 能进行各种逻辑运算：与或非

- 对应代码：
  - nemu/include/cpu/alu.h
  - nemu/src/cpu/alu.c

算术逻辑单元

- 取add为例

函数名称，对应指令名称　　参与运算的两个操作数　　操作数长度：8, 16, 32

```
uint32_t alu_add(uint32_t src, uint32_t dest, size_t data_size)
{
        printf("\e[0;31mPlease implement me at alu.c\e[0m\n");
        assert(0);
        return 0;
}
```

要替换成教程中说明的正确实现：

1. 返回dest + src的结果，data_size不足32时，高位清0
2. 设置EFLAGS标志位

实现对一种运算的模拟

第一步：找到需要实现的函数

执行make test_pa-1遇到错误提示

```
pa2020_spring$ make clean
pa2020_spring$ make test_pa-1

./nemu/nemu --test-reg
NEMU execute built-in tests
reg_test()       pass

./nemu/nemu --test-alu add
NEMU execute built-in tests
Please implement me at alu.c
```

需要实现add

实现对一种运算的模拟

第二步：掏出i386手册

1. 找到i386手册Sec. 17.2.2.11
2. 有关ADD指令的描述 p.g. 261 of 421
3. 看Flags Affected: OF, SF, ZF, AF, CF, and PF as described in Appendix C
4. 找到Appendix C并仔细体会（AF不模拟）

实现对一种运算的模拟

第三步：按照手册规定的操作进行实现

nemu/src/cpu/alu.c
add的参考实现方案

```
uint32_t alu_add(uint32_t src, uint32_t dest, size_t data_size) {
	uint32_t res = 0;
	res = dest + src;                       // 获取计算结果

	set_CF_add(res, src, data_size);        // 设置标志位
	set_PF(res);
	// set_AF();                            // 我们不模拟AF
	set_ZF(res, data_size);
	set_SF(res, data_size);
	set_OF_add(res, src, dest, data_size);

	return res & (0xFFFFFFFF >> (32 - data_size)); // 高位清零并返回
}
```

```
// CF contains information relevant to unsigned integers
void set_CF_add(uint32_t result, uint32_t src, size_t data_size) {
        result = sign_ext(result & (0xFFFFFFFF >> (32 - data_size)), data_size);
        src = sign_ext(src & (0xFFFFFFFF >> (32 - data_size)), data_size);
        cpu.eflags.CF = result < src; //
}
```

对cpu中eflags寄存器的访问
i386手册 sec 2.3.4.1

adc的CF要如何判断？ dest + src +CF

```
void set_ZF(uint32_t result, size_t data_size) {
        result = result & (0xFFFFFFFF >> (32 - data_size));
        cpu.eflags.ZF = (result == 0);
}

// SF and OF contain information relevant to signed integers
void set_SF(uint32_t result, size_t data_size) {
        result = sign_ext(result & (0xFFFFFFFF >> (32 - data_size)), data_size);
        cpu.eflags.SF = sign(result);
}

void set_PF(uint32_t result) { … } // 简单暴力穷举也行
```

nemu/src/cpu/alu.c
add的参考实现方案

```
void set_OF_add(uint32_t result, uint32_t src, uint32_t dest, size_t data_size) {
	switch(data_size) {
		case 8:
			result = sign_ext(result & 0xFF, 8);
			src = sign_ext(src & 0xFF, 8);
			dest = sign_ext(dest & 0xFF, 8);
			break;
		case 16:
			result = sign_ext(result & 0xFFFF, 16);
			src = sign_ext(src & 0xFFFF, 16);
			dest = sign_ext(dest & 0xFFFF, 16);
			break;
		default: break;// do nothing
	}
	if(sign(src) == sign(dest)) {
		if(sign(src) != sign(result))
			cpu.eflags.OF = 1;
		else
			cpu.eflags.OF = 0;
	} else {
		cpu.eflags.OF = 0;
	}
}
```

nemu/src/cpu/alu.c
add的参考实现方案

实现对一种运算的模拟

重复第一步：找到需要实现的函数

执行make test_pa-1遇到错误提示

```
./nemu/nemu --test-alu add
NEMU execute built-in tests
alu_test_add()  pass
./nemu/nemu --test-alu adc
NEMU execute built-in tests
Please implement me at alu.c
```

需要实现adc

实现对ALU的模拟

完成对ALU的模拟

```
alu_test_add()  pass
alu_test_adc()  pass
alu_test_sub()  pass
alu_test_sbb()  pass
alu_test_and()  pass
alu_test_or()   pass
alu_test_xor()  pass
alu_test_shl()  pass
alu_test_shr()  pass
alu_test_sal()  pass
alu_test_sar()  pass
alu_test_mul()  pass
alu_test_div()  pass
alu_test_imul() pass
alu_test_idiv() pass
```

测试用例代码:

nemu/src/cpu/test/alu_test.c

注意: 移位操作不测试OF位
所有操作都不测试AF位
imul操作所有标志位都不测试

注意: 禁止采用测试用例里面使用内联汇编进行实现的方法，但是可以学习这种交叉验证的方法

说明: 其中mod和imod运算是我们单独抽象出来的，需参照div和idiv的说明进行实现并测试

重要说明

特别说明：针对上面四个移位操作，约定只影响 `dest` *操作数的低* `data_size` *位，而不影响其高* `32 - data_size` *位。标志位的设置根据结果的低* `data_size` *位来设置。*

——感谢16级何峰彬、张明超同学的建议

该特别说明不再生效，移位操作也需要将高位清零

实现对ALU的模拟

- 实现ALU的目的
  - 复习课本第二章内容
  - 在alu.c中实现的这些函数，到了PA 2实现对应指令的时候，就可以直接调用了
- PA 1不设置小的阶段截止，PA 1-3完成后统一提交
  - 建议PA 1-1和PA 1-2在一周内完成

PA 1-2 结束