

2023 春季学期-计算机组成原理-作业二

(要求：独立完成，4 月 23 日上课时上交)

1. (10 points) (课本 P125 页习题 4) 假设地址为1200H的内存单元中的内容为12FCH，地址为12FCH的内存单元的内容为38B8H，而38B8H单元的内容为88F9H。说明以下各情况下操作数的有效地址是多少？

- (1) 操作数采用变址寻址，变址寄存器的内容为 252，指令中给出的形式地址为1200H。
- (2) 操作数采用一次间接寻址，指令中给出的地址码为1200H。
- (3) 操作数采用寄存器间接寻址，指令中给出的寄存器编号为 8,8 号寄存器的内容为1200H。

(1) 变址寻址 = 基准地址 + 变址寄存器，因此操作数有效地址为 $1200H + FCH = 12FCH$ 。

(2) 地址码为存放有效地址的主存单元地址，1200H 对应的内容为 12FCH。

(3) 寄存器中存放的是操作数有效地址，有效地址为 1200H。

2. (10 points) (课本 P125 页习题 6) 某计算机指令系统采用定长指令字格式，指令字长 16 位，每个操作数的地址码长 6 位。指令分二地址、单地址和零地址 3 类。若二地址指令有 k_2 条，零地址指令有 k_0 条，则单地址指令最多有多少条？

- 二地址指令操作码为 4 位，因此，有 16 种操作码。
- 题目中给出二地址指令有 k_2 条，因此有 $16 - k_2$ 种组合未被设定为二地址指令的操作码。它们可被一地址指令使用。
- 一地址指令操作码有 10 位，其中，高 4 位为二地址指令中未使用的几种组合。因此，对于一地址格式，有 $(16 - k_2) \times 2^6$ 种操作码。由于还有零地址指令，因此，这些操作码不能全部被一地址指令占用。
- 假设一地址指令共有 k_1 条，则根据同样的思路，零地址指令操作码数量可以表示为 $((16 - k_2) \times 2^6 - k_1) \times 2^6$ 。由于总共三种类型指令已经全部考虑在内，因此，上述表达式的值等于 k_0 。
- 由此可求得 $k_1 = (16 - k_2) \times 2^6 - k_0 / 2^6$

3. (20 points) (课本 P125 页习题 8) 某计算机字长为 16 位，主存地址空间大小为 128KB，按字编址。采用单字长定长指令格式，指令各字段定义如下：



转移指令采用相对寻址方式，相对位移量用补码表示，寻址方式定义如下表所示。

请回答下列问题：

- (1) 该指令系统最多可有多少条指令？最多有多少个通用寄存器？存储器地址寄存器（MAR）和存储器数据寄存器（MDR）至少各需要多少位？

Ms/Md	寻址方式	助记符	含义
000B	寄存器直接	Rn	操作数 = R[Rn]
001B	寄存器间接	(Rn)	操作数 = M[R[Rn]]
010B	寄存器间接、自增	(Rn)+	操作数 = M[R[Rn]], R[Rn] ← R[Rn] + 1
011B	相对	D(Rn)	转移目标地址 = PC + R[Rn]

(2) 转移指令的目标地址范围是多少？

(3) 若操作码0010B表示加法操作（助记符为add），寄存器R4和R5的编号分别为100B和101B，R4的内容为1234H，R5的内容为5678H，地址1234H中的内容为5678H，地址5678H中的内容为1234H，则汇编语句add(R4), (R5)+（逗号前为第一源操作数，逗号后为第二源操作数和目的操作数）对应的机器码是什么（用十六进制表示）？该指令执行后，哪些寄存器和存储单元的内容会改变？改变后的内容是什么？

(1) 采用单字长指令格式，操作码占 4 位，所以最多有 16 条指令。

指令中通用寄存器编号占 3 位，所以，最多有 8 个通用寄存器。

地址空间位 128KB，按字编址，所以总共有 64K 个存储单元，因此，地址位数为 16 位。所以，MAR 至少为 16 位。因为字长为 16 位，所以 MDR 至少为 16 位。

(2) 地址位数和字长都是 16 位，所以 PC 和通用寄存器位数都是 16，两个 16 位数相加，结果至少 16 位，所以转移目标地址位数为 16，因此能在整个地址空间转移。所以目标转移地址范围是 0000H-FFFFH。

(3) 对于add (R4), (R5)+，将指令代码各字段拼接起来就能够得到其机器码。

- add 对应 op 字段，为 0010B
- (R4) 的寻址方式字段为 001B，R4 编号为 100B
- (R5)+ 的寻址方式字段为 010B，R5 编号为 101B
- 因此机器码为 0010_001100_010101，即 2315H。

指令功能为 $M[R5] \leftarrow M[R[R5]] + M[R[R4]]$, $R[R5] \leftarrow R[R5] + 1$ 。

- 已知 $R[R4] = 1234H$, $R[R5] = 5678H$.
- $M[R[R4]] = M[1234H] = 5678H$, $M[R[R5]] = M[5678H] = 1234H$.
- 运算结果为 $1234H + 5678H = 68ACH$.
- 所以，地址 5678H 中的内容从 1234H 变为 68ACH，同时 R5 内容从 5678H 变为 5679H。

4. (10 points) (课本 P126 页习题 10) 以下程序段是某个过程对应的指令序列。入口参数int a和int b分别置于\$a0和\$a1中，返回参数是该过程的结果，置于\$v0中。要求为以下 MIPS 指令序列加注释，并简单说明该过程的功能。

```

    add $t0, $zero, $zero
loop: beq $a1, $zero, finish
    add $t0, $t0, $a0

```

```

        sub $a1, $a1, 1
        j loop
finish: addi $t0, $t0, 100
        add $v0, $t0, $zero

```

```

        add $t0, $zero, $zero    # 寄存器$t0置0
loop:   beq $a1, $zero, finish    # 如果$a1的值为0, 转入finish
        add $t0, $t0, $a0        # 将$t0和$a0的内容相加, 送到$t0
        sub $a1, $a1, 1         # 将$a1的值减1
        j loop                  # 无条件转到loop处
finish: addi $t0, $t0, 100       # 将$t0的内容加100
        add $v0, $t0, $zero     # 将$t0的内容送到$v0

```

该过程计算 $a \times b + 100$ 。

5. (14 points) (课本 P126 页习题 11) 下列指令序列用来对两个数组进行处理, 并产生结果存放在 \$v0 中, 两个数组的基地址分别存放在 \$a0 和 \$a1 中, 数组长度分别存放在 \$a2 和 \$a3 中。要求为以下 MIPS 指令序列加注释, 并简单说明该过程的功能。假定每个数组有 2500 个字, 其数组下标为 0-2499, 该指令序列运行在一个时钟频率为 2GHz 的处理器上, add、addi 和 sll 指令的 CPI 为 1; lw 和 bne 指令的 CPI 为 2, 则最坏情况下运行所需时间是多少秒?

```

        sll $a2, $a2, 2
        sll $a3, $a3, 2
        add $v0, $zero, $zero
        add $t0, $zero, $zero
outer:  add $t4, $a0, $t0
        lw $t4, 0($t4)
        add $t1, $zero, $zero
inner:  add $t3, $a1, $t1
        lw $t3, 0($t3)
        bne $t3, $t4, skip
        addi $v0, $v0, 1
skip:   addi $t1, $t1, 4
        bne $t1, $a3, inner
        addi $t0, $t0, 4
        bne $t0, $a2, outer

```

```

        sll $a2, $a2, 2    # $a2的内容左移两位
        sll $a3, $a3, 2    # $a3的内容左移两位
        add $v0, $zero, $zero # $v0置0
        add $t0, $zero, $zero # $t0置0
outer:  add $t4, $a0, $t0    # $a0加$t0送到$t4, 即计算数组1当前元素的地址
        lw $t4, 0($t4)     # 数组1当前元素载入$t4
        add $t1, $zero, $zero # $t1置0
inner:  add $t3, $a1, $t1    # $a1加$t1送到$t3, 即计算数组2当前元素的地址

```

```

lw $t3, 0($t3)      # 数组2当前元素载入$t3
bne $t3, $t4, skip  # $t3和$t4内容不相等，则转入skip
addi $v0, $v0, 1     # $v0加1
skip: addi $t1, $t1, 4 # $t1加4，地址增量
bne $t1, $a3, inner  # $t1和$a3不相等，即数组2未处理完，则转入inner执行
addi $t0, $t0, 4     # $t0加4，地址增量
bne $t0, $a2, outer  # $t0和$a2不相等，即数组1未处理完，则转入outer执行

```

- 该程序的功能为统计两个数组中相同元素的个数。
- 最坏的情况是两个数组完全相同，这样的话，指令 `addi $v0, $v0, 1` 每次都被执行。
- 最坏情况下，总的时钟周期数可以通过 CPI 计算，结果为 56267506。
- 总的运行时间为 $56267506 \times 0.5\text{ns} \approx 0.028\text{s}$ 。

6. (10 points) (课本 P127 页习题 17) 假定编译器将 `a` 和 `b` 分别分配到 `t0` 和 `t1` 中，用一条 RV32I 指令或最短的 RV32I 指令序列实现以下 C 语言语句：`b=31&a`。如果把 31 换成 65535，即 `b=65535&a`，则用 RV32I 指令或指令序列如何实现？对比 RV32I 与 MIPS 之间在立即数处理上有何不同？

实现 `b=31&a` 只需要一条 RV32I 指令：`andi $t1, $t0, 31`。

RISC-V 中 `andi` 指令为 I 型指令，其立即数部分为 12 位。而 $65535 = 1111_1111_1111_1111\text{B}$ 无法用 12 位整数表示。可以用以下指令实现 `b=65535&a`：

$65535 = 0000_0000_0000_0000_1111_1111_1111_1111$

`lui t1, 15` 将 65535 的高 20 位存入 `t1` 的高 20 位中，并将低 12 位清零。

`addi t1, t1, 4095` 将 65535 的低 12 位加到 `t1` 中。

`and t1, t1, t0` 进行逻辑与操作。

MIPS 指令集中 I 型指令立即数为 16 位，而 RISC-V 中，I 型指令的立即数为 12 位。在使用 `lui` 载入时，MIPS 将立即数载入到寄存器的高 16 位，而 RISC-V 将立即数载入到寄存器的高 20 位，且 `lui` 为 U 型指令。