日期: /

3、21

【3】 1). addi x_0, x_0, 0

2). jalr x_0, ra, 0

3). auipc a1, offset[31:12]

jalr x_0, a1, offset[11:0]

4). addi rd, rs, 0

5). csrrs rd, cycle, x_0

6). addiw rd, rs, 0

【7】

1). slt t3, t0, t2

slt t4, t0, t1

2). add t0, t1, t2

slt t3, t0, t1

bne x_0, t3, overflow

3). X86: 存在标志寄存器，其中就有进位标志 CF、溢出标志 OF

加法有符号溢出时，OF置1；无符号溢出时，CF置1

ARM: 和X86类似，存在 CPSR，存储进位标志、溢出标志

MIPS: 用异常或 HI/LO 寄存器来处理，ADD溢出则异常，

ADDU溢出则将结果写入 rd

【8】

D. DIVU REMU DIV REM

$2^{XLEN}-1$ x -1 x

日期: /

目的：为了简化除法电路，于是让所有除0都返回，从而避免控制流更改

2、NV：无效操作

DZ：除数为0

OF：溢出(向上)

UF：向下溢出

NX：不精确的。

不会陷入系统调用

3)、X86：触发DE异常，转移控制权到异常处理程序，将异常号、错误代码压入堆栈

ARM：触发"Divide by Zero"异常，转移控制权到异常处理程序，将异常状态信息存储在特殊处理器

MIPS：触发"Divide by Zero"异常，转移控制权到异常处理程序，将异常号和附加信息存储在特殊处理器

12、1)、M态或S态，通常S态　　2)、M态　　3)、S态

4)、S态　　5)、U态

13、

VecMUL：

```
addi sp, sp, -32
sw   a0, 28(sp)
sw   s0, 24(sp)
```

日期: /

```
addi s0, sp, 32
addi a5, x0, 0
addi a7, a0, 0
part1:
lw a3, 0(t1)
mul a4, a3, t2
sw a4, 0(a7)
addi t1, t1, 1
addi a7, a7, 4
addi a5, a5, 4
slti a6, a5, 99
bne a6, x0, part1
lw a0, 0(t0)
lw ra, 28(sp)
lw s0, 24(sp)
addi sp, sp, 32
ret
```

14.

```
part1: ble a0, a1, part2
       add a2, a1, a0
part2: sub a2, a0, a1
```

日期: /

15.

```
sw   t0, 0(t0)
addi t1, x0, 3
addi t2, t0, 4
sw   t1, 0(t2)
slli t2, t1, 2
add  t2, t0, t2
sw   t1, 0(t2)
```

17.

a1 左移 30 位并返回

同时返回 $a_0 = 30$

16.

```
swap: addi sp, sp, -32
      sw   ra, 28(sp)
      sw   s0, 24(sp)
      addi s0, sp, 32
      lw   t2, 0(t0)
      lw   t3, 0(t1)
      sw   t3, 0(t0)
      sw   t2, 0(t1)
      lw   ra, 28(sp)
      lw   s0, 24(sp)
      addi sp, sp, 32
      ret
```