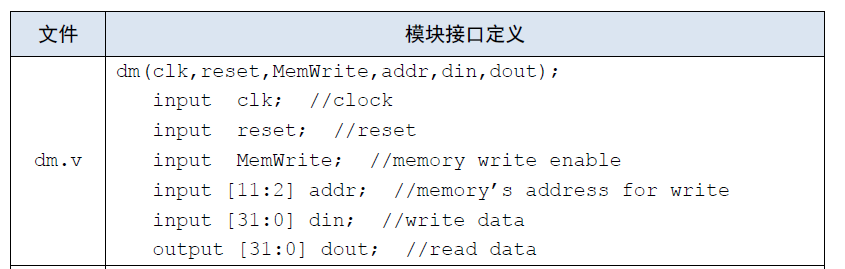
1. 根据你的理解，在下面给出的DM的输入示例中，地址信号addr位数为什么是[11:2]而不是[9:0]？这个addr信号又是从哪里来的？



答：addr来自于alu计算得到的写入地址，由于此时定义DM大小仅为1024字，加之写入地址为字节地址，所以去掉两位（除以四）后再取十个二进制位

1. 在相应的部件中，**reset的优先级**比其他控制信号（不包括clk信号）都要**高**，且相应的设计都是**同步复位**。清零信号reset是针对哪些部件进行清零复位操作？这些部件为什么需要清零？

答：PC,GRF,DM

PC复位是为了将指令地址初始化，包括上电之初

GRF复位是因为每次上电运行，寄存器种初始值都应该是零而不是之前的旧值

DM复位同理

1. 列举出用Verilog语言设计控制器的几种编码方式（至少三种），并给出代码示例。

答：

1. 用case/if-else语句

if (Op == 6'b000100&&Eq == 1'b1)

NPC\_ctrl = 2'b01;

else if (Funct == 6'b001000&&Op == 6'b000000)

NPC\_ctrl = 2'b10;

else if (Op == 6'b000011)

NPC\_ctrl = 2'b11;

else

NPC\_ctrl = 2'b00;

case(Op)

//R-Type

6'b000000: begin

case (Funct)

//addu

6'b100001: signal <= 10'b0000100000;

//subu

6'b100011: signal <= 10'b0000100010;

//jr

6'b001000: signal <= 10'b0000000000;

default: signal <= 10'b0000000000;

endcase

end

//ori

6'b001101: signal <= 10'b1000101110;

//lw

6'b100011: signal <= 10'b1001111000;

//sw

6'b101011: signal <= 10'b0000011001;

//beq

6'b000100: signal <= 10'b0000010100;

//lui

6'b001111: signal <= 10'b1010100000;

//jal

6'b000011: signal <= 10'b0111100000;

//including nop

default: signal <= 10'b0000000000;

endcase

1. 用assign语句

assign sw = (Funct == 6’b101011)

……

assign RegWr = ~(sw&&beq&&jr)

assign MemWr = sw

（3） 用宏定义

‘sw Funct == 6’b101011

‘beq Funct == 6’b000100

……

‘RegWr ~(sw&&beq&&jr)

‘MemWr sw

1. 根据你所列举的编码方式，说明他们的优缺点。

答：

1. Case和if-else语句优点在于情况划分清晰，易于调试，逻辑一目了然；

但是缺点在于一旦语句块内部逻辑复杂，分支众多时，会难以调试且容易产生bug

1. assign语句描述组合逻辑，速度快且简单明了；

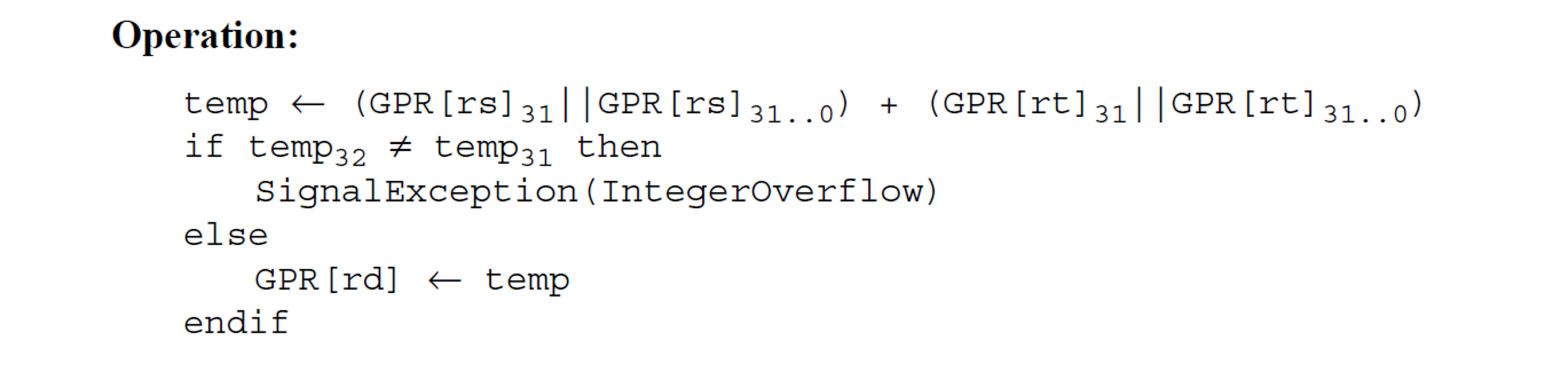
但用来产生控制信号时，逻辑表达式可能会比较冗长，且中间容易出错。

1. 宏定义优点在于提高了程序可读性，方便进行修改；提高了运行效率，运用带参数宏定义可以完成某些小模块的功能；

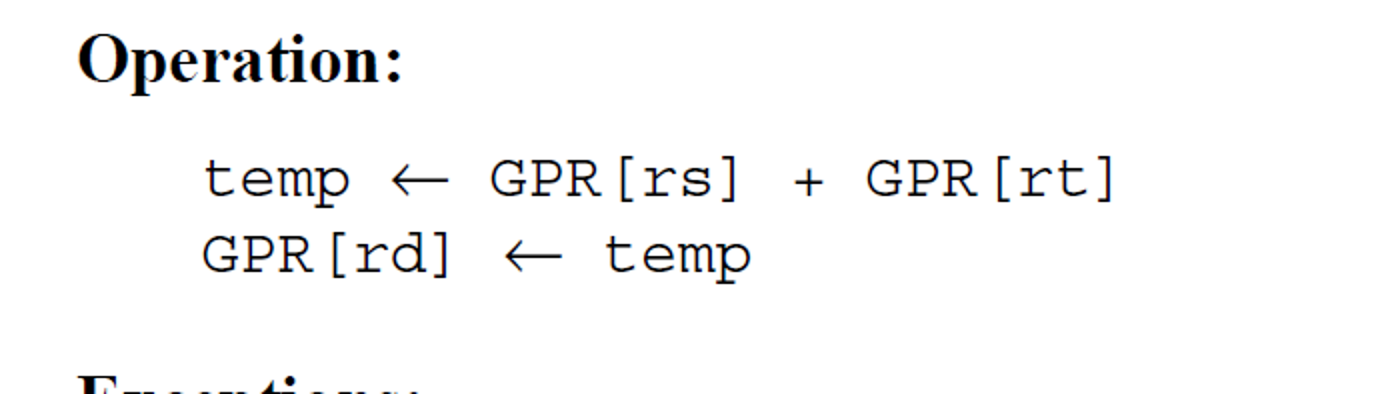
缺点在于比如上面例子里嵌套使用了‘sw，当嵌套过多时影响可读性容易出错；带参数宏定义直接替换，不检查参数合法性和类型，可能引发一系列错误；

1. C语言是一种弱类型程序设计语言。C语言中不对计算结果溢出进行处理，这意味着C语言要求程序员必须很清楚计算结果是否会导致溢出。因此，如果仅仅支持C语言，MIPS指令的所有计算指令均可以忽略溢出。 请说明为什么在忽略溢出的前提下，addi与addiu是等价的，add与addu是等价的。提示：阅读《MIPS32® Architecture For Programmers Volume II: The MIPS32® Instruction Set》中相关指令的Operation部分 。

答： 下图是手册中对ADD进行的操作解释，和ADDU对比，唯一区别就在于，后者不比较额外添加的32位是否和31位相同，其余加法操作均一样，因此在忽略溢出时，二则等价，同理适用于addi和addiu



Add



Addu

1. 根据自己的设计说明单周期处理器的优缺点。

答：

优点：稳定，相比较流水线，不存在冒险情况，且数据通路主要为组合逻辑，根据真值表即可确定输出；简单，无论是设计调试还是模拟仿真，单周期在逻辑上都更加简单，易于理解，而且某种程度上越简单可靠性越高，因此单周期也更加可靠；

缺点：时钟周期由指令集中最慢的指令决定（lw），浪费性能；一个周期只能执行一条指令，速度太慢。

1. 简要说明jal、jr和堆栈的关系。

答：在MIPS处理器中主要参与计算的是寄存器，在一条32位指令里不足以包含有效的内存地址，所以借助$ra来实现函数的返回。几乎在每个函数调用中都会使用到这个寄存器，因此$ra会被保存在堆栈上以避免被后面的函数调用修改，当函数需要返回时，从堆栈上取回$ra然后跳转。

当进入函数时，jal (jump and link) 跳转到指定标签（jump）并且将PC+4存入$ra以备返回（link）。进入函数后如果会修改$ra则压栈，函数调用结束时先出栈再jr $ra(jump register)跳转回存储在$ra中的PC+4（相对于函数调用指令）