

4.21.1

指令个数 (没有旁路) =  $n + 0.4n = 1.4n$ 

时钟周期 = 250ps

 $\therefore CPI = 1$ 所以  $T = 1 \times 1.4n \times 250ps$ 指令个数 (有旁路) =  $n + 0.05n = 1.05n$ 

时钟周期 = 300ps

 $\therefore CPI = 1$ 所以  $T = 1 \times 1.05n \times 300$  $\therefore$  加速比 =  $\frac{1.4 \times 250}{1.05 \times 300} = 1.1111$ 

4.21.2

设带旁路之后的 nop 指令占指令的百分之  $x$ ，所以 nop 指令的数量为  $x \cdot n$ ，在程序设计中应该要满足下面不等式：

$300 \times x(1+x)n < 250 \times 1.4n$ ，  
也就是带旁路之后的应该要比带旁路之前要快，所以  
 $x < 16.7\%$



# 南开大学 作业纸

系别\_\_\_\_\_ 班级\_\_\_\_\_ 姓名\_\_\_\_\_ 第 页

4.23.3

由题意可知, 带旁路之前的 nop 数量为  $n \cdot x$ ,  
所以 我们设带旁路之后的 nop 占全部指令的百分之  $\eta$ ,  
所以 满足下面不等式:

$$300 \times (1 + \eta) \times n < 250 \times (1 + x) \times n$$

$$1 + \eta < \frac{5}{6} (1 + x)$$

$$\eta < \frac{5}{6} x - \frac{1}{6}$$

4.24

不会更快.

$$T(\text{没有旁路}) = 250 \times (1 + 0.075) n = 268.75 n$$

$$T(\text{有旁路}) > 300 \times n = 300 n$$

所以加上旁路反而会变得更慢.

4.21.5

我们可以看 4.22.3 中的例子:

$$300 \times (1 + \eta) \times n < 250 \times (1 + x) \times n$$

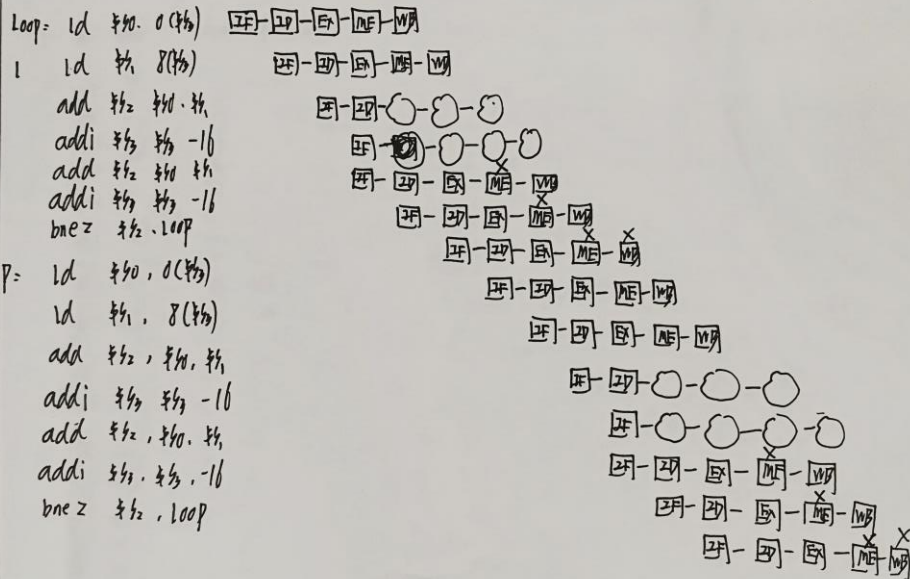
$$\text{至少要满足 } 300 n < 250 \times (1 + x) \times n$$

$$\therefore x > \frac{1}{5} \quad \text{所以至少要 } 0.2n \text{ 条 nop.}$$



4.25.1.

因为在 add 指令中要用到 ld 指令的结果所以会产生阻塞。



4.25.2

我们知道 add 指令和 addi 指令并不使用数据寄存器, bnez 指令也并不使用数据寄存器, 也不需写回寄存器堆, 所以相应的流水级就是无用的, 在 4.25.1 中已经用 "X" 标记。

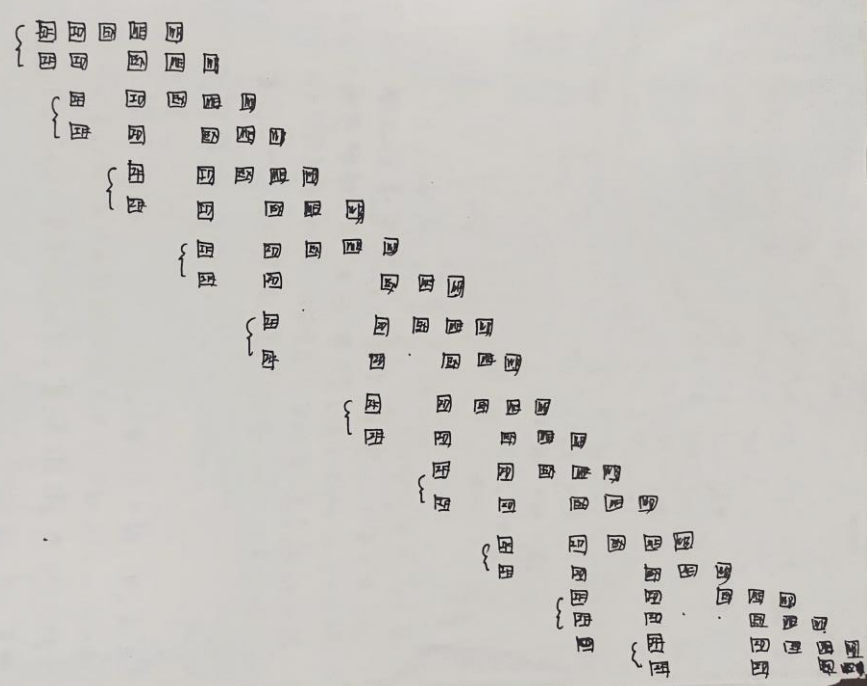
标记后, 我们可以发现, 并不存在某一个时钟周期, 五个流水级全部在做有用功, 或者是被阻塞了, 或者在做无用功。



南京大学作业纸

系别 \_\_\_\_\_ 班级 \_\_\_\_\_ 姓名 \_\_\_\_\_ 第 \_\_\_\_\_ 页

4.3.1.





# 南开大学 作业纸

系别 \_\_\_\_\_ 班级 \_\_\_\_\_ 姓名 \_\_\_\_\_ 第 \_\_\_\_\_ 页

4.3.2

我们知道在单发射处理器中这个循环需要10个时钟周期。我们也可从4.3.1中看出在双发射处理器中这个循环每次也需要10个时钟周期，所以并没有加速。

4.3.3

我们从原来的代码中可以看出，变量*i*保存在*ra*中，所以我们可以一开始就判断*i*是否等于0。如果等于零直接跳出循环。然后我们可以知道原来的代码把 *i=i+2* 放到了最后面，但通过分析代码可以知道，这个代码完全可以早点执行。这样可以把 *sub* 指令延迟一个时钟周期，正好可以避免一个阻塞。

```
beqz ra, next
li ra, 0
jal ENT
Top: slli ra, ra, 3
add ra, ra, ra
lw ra, 0(ra)
lw ra, 8(ra)
addi ra, ra, 2
sub ra, ra, ra
add ra, ra, ra
sw ra, 0(ra)
ENT: bne ra, ra, Top
next:
```





# 南 京 大 学 作 业 纸

系别 \_\_\_\_\_ 班级 \_\_\_\_\_ 姓名 \_\_\_\_\_ 第 页

4.3.4.

再次分析代码后, 可以发现, 计算 `brl` 的地址的代码也可以早一点计算, 不用到那么晚才执行.

```
beqz $s1, next
li $t0, 0
Top = slli $t0, $t0, 3
add $t1, $t2, $t0
lw $t2, 0($t1)
add $t5, $t3, $t0
lw $t3, 8($t1)
sub $t4, $t2, $t3
addi $t0, $t0, 2
sw $t4, 0($t5)
bne $t0, $s1, Top
next:
```

4.3.5

beqz \$s1, next	IF-ID-EX-ME-WB
li \$t0, 0	IF-ID-EX-ME-WB
slli \$t0, \$t0, 3	IF-ID-EX-ME-WB
add \$t1, \$t2, \$t0	IF-ID-EX-ME-WB
lw \$t2, 0(\$t1)	IF-ID-EX-ME-WB
add \$t5, \$t3, \$t0	IF-ID-EX-ME-WB
lw \$t3, 8(\$t1)	IF-ID-EX-ME-WB
addi \$t0, \$t0, 2	IF-ID-EX-ME-WB
sub \$t4, \$t2, \$t3	IF-ID-EX-ME-WB
sw \$t4, 0(\$t5)	IF-ID-EX-ME-WB
bne \$t0, \$s1, Top	IF-ID-EX-ME-WB
slli \$t0, \$t0, 3	IF-ID-EX-ME-WB
add \$t1, \$t2, \$t0	IF-ID-EX-ME-WB
lw \$t2, 0(\$t1)	IF-ID-EX-ME-WB
add \$t5, \$t3, \$t0	IF-ID-EX-ME-WB
lw \$t3, 8(\$t1)	IF-ID-EX-ME-WB
addi \$t0, \$t0, 2	IF-ID-EX-ME-WB
sub \$t4, \$t2, \$t3	IF-ID-EX-ME-WB
sw \$t4, 0(\$t5)	IF-ID-EX-ME-WB
bne \$t0, \$s1, Top	IF-ID-EX-ME-WB



# 南开大学 作业纸

系别 \_\_\_\_\_ 班级 \_\_\_\_\_ 姓名 \_\_\_\_\_ 第 \_\_\_\_\_ 页

4.3.6

通过上面的分析和画图可知, 4.3.3 中的代码每个循环差不多需要 9 个时钟周期, 而 4.3.4 的代码需要 7.5 个时钟周期, 所以加速比为:  $\frac{9}{7.5} = 1.2$

4.3.7

因为要在每个循环中实现原先的两次迭代, 所以需要四次访存, 两次减法并需要把  $z$  每次加 4.

beqz. \$s1, next

li \$s0, 0.

Top: slli. \$t0, \$s0, 3

add \$t1, \$t2, \$t0

add \$t5, \$t3, \$t0

lw \$t2, 0(\$t1)

lw \$t3, 8(\$t1)

lw \$t4, 16(\$t1)

lw. \$t6, 24(\$t1)

addi. \$s0, \$s0, 4

sub. \$t7, \$t2, \$t3

sub. \$t8, \$t4, \$t6

sw \$t7, 0(\$t5)

sw \$t8, 16(\$t5)

bne. \$s0, \$s1, Top.

next:



# 南开大学 作业纸

系别 \_\_\_\_\_ 班级 \_\_\_\_\_ 姓名 \_\_\_\_\_ 第 \_\_\_\_\_ 页

4.31.8

beqz.  $r_1$ . next

li  $r_0$ , 0

addi  $r_1$ ,  $r_2$ ,  $r_{zero}$

Top:

lw.  $r_2$  0( $r_1$ )

add.  $r_5$ ,  $r_2$ ,  $r_0$

lw.  $r_3$  8( $r_1$ )

addi  $r_0$ ,  $r_0$ , 4

lw.  $r_4$  16( $r_1$ )

slli  $r_0$ ,  $r_0$ , 3

lw  $r_6$  24( $r_1$ )

sub.  $r_7$ ,  $r_2$ ,  $r_3$

sw  $r_7$ , 0( $r_5$ )

sub  $r_8$ ,  $r_4$ ,  $r_6$

sw  $r_8$  16( $r_5$ )

add  $r_1$ ,  $r_2$ ,  $r_0$

bne  $r_0$ ,  $r_1$ , Top

next:

4.31.9


4.31.7中的代码每次执行需要13个周期  
也就是说每一次迭代需要6.5个时钟周期。

而4.31.8中的代码每次执行需要7.5个周期  
也就是每一次迭代需要3.75个时钟周期。

所以加速比为  $\frac{6.5}{3.75} \approx 1.7$



4.31.10	
beqz \$a, next	IF ID EX ME WB
li \$s0, 0	IF ID <del>EX</del> ME WB
addi \$t1, \$s2, \$s0	IF ... ID EX ME WB
lw \$t2 (\$t1)	IF ... ID ... EX ME WB
add \$t5, \$t2, \$t1	IF ... ID EX ME WB
lw \$t3, 8(\$t1)	ZF ... ID EX ME WB
<del>addi \$t0, \$s0, 3</del>	
addi \$s0, \$s0, 4	ZF ID EX ME WB
lw \$t4, 16(\$t1)	ZF ID EX ME WB
slli \$t0, \$s0, 3	IF ID EX ME WB
ld \$t0, 24(\$t1)	ZF ID EX ME WB
sub \$t7, <del>\$s0</del> \$t0, \$t3	ZF ID EX ME WB
sw \$t7, 0(\$t5)	ZF ID ... EX ME WB
sub \$t8, \$t7, \$t2	ZF ... ID EX ME WB
sw \$t7, 0(\$t5)	ZF ... ID EX ME WB
add \$t1, \$t2, \$t0	ZF ID EX ME WB
bne \$s0, \$t1, top	ZF ID EX ME WB
lw \$t2 (\$t1)	ZF ID EX ME WB
add \$t5, \$t2, \$t1	ZF ID EX ME WB
lw \$t3, 8(\$t1)	IF ID EX ME WB
addi \$s0, \$s0, 4	ZF ID EX ME WB
lw \$t4, 16(\$t1)	IF ID EX ME WB
slli \$t0, \$s0, 3	ZF ID EX ME WB
ld \$t0, 24(\$t1)	


南开大学 作业纸

4.31.10

如果使用与4.31.8中相同的代码,因为没有出现由结构冒险所引发的停顿,所以改进之后的双发射处理器并不会提高程序性能。



# 南开大学 作业纸

系别 \_\_\_\_\_ 班级 \_\_\_\_\_ 姓名 \_\_\_\_\_ 第 \_\_\_\_\_ 页

4.32.1

在流水线和单周期中实现一条加法指令所消耗的能量相同 =  $140 + 70 \times 2 + 60 = 340 \text{ pJ}$

因为加法要读两个寄存器并写入一个寄存器并且不访问数据存储器。

4.32.2

因为 lw 要访问全部阶段，所以自然而然它所消耗的能量最多。它要读一个寄存器，写入一个寄存器，并且要读数据存储器，所以消耗的总能量为： $140 + 70 \times 4 + 60 + 140 = 480 \text{ pJ}$

4.32.3

我们在 4.32.2 中可以发现，lw 指令实际中读一个寄存器即可，但因为流水线的设计我们还读了 rt，这个能量我们可以不用消耗，所以我们在设计流水线时可以先判断是不是 lw 指令，如果是的话就可以不用再读第二个寄存器。这样，我们就可以减少 70 pJ 的消耗。能耗降低的比例为： $\frac{70}{480} = 14.6\%$



# 南开大学 作业纸

系别 \_\_\_\_\_ 班级 \_\_\_\_\_ 姓名 \_\_\_\_\_ 第 \_\_\_\_\_ 页

4.32.4

我们不仅限于lw指令, 4.32.3中的改进对sw指令和各种I型指令都有潜在收益, 因为它们的第二个操作数都来自于低16位, 而不是rt寄存器。我们可以接着思考, 我们再改进一点, 不只是针对rt寄存器, 而是针对rs, rt两个, 会发现对无条件跳转指令JMP也有收益。

4.32.5

在改进之前, 控制单元和寄存器读取是同步进行的, 但现在, 控制单元要首先产生一个控制信号来控制寄存器堆要不要读取寄存器。意味着, 控制单元和寄存器堆不能重叠, 所以这个阶段的总时间就变成了  $150 + 90 = 240ps$ 。但还是小于最长的阶段的250ps, 所以对CPU性能无影响。

4.32.6

能正常工作是因为有各类其它控制信号和多路选择器的存在, 因为存在这些, 保证了从数据存储器中取到的数据不会乱放入寄存器或别的地方。

这种改变对时钟周期不会有影响, 因为时钟周期本来就是满足全部阶段。但是这种改变会影响能耗, 每次都要访问数据存储器意味着除了lw的每条指令都将额外增加140pJ的能耗。