# **组合逻辑作业**

1. 化简逻辑表达式：。

答案：

过程：

1. 化简逻辑表达式

答案：

过程：

分为两部分：1）和2）

= 0

合并：

1. 化简逻辑表达式：

答案：

过程：

1. 化简逻辑表达式：

答案：

过程：

1. 化简逻辑表达式：

答案：先完成；卡诺图（注意要把ABC展开成4个由5变量组成的乘积项）；最简表达式为

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ABC  DE | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
| 00 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 01 | 1 |  |  |  | 1 | 1 | 1 | 1 |
| 11 | 1 |  |  |  | 1 | 1 | 1 | 1 |
| 10 | 1 |  |  |  | 1 | 1 | 1 | 1 |

【注意】000列是不能与111、101、100构成4列的！！！

过程：

Column code: COL-ROW

ABC -> 6-1，6-2，6-3，6-4

ABD -> 5-3，5-4，6-3，6-4

ABE -> 5-2，5-3，6-2，6-3

ACD -> 6-3，6-4，7-3，7-4

ACE -> 6-2，6-3，7-2，7-3

~（ADE） -> 1-1，2-1，3-1，4-1

~B~CD -> 1-1，1-2，1-3，8-1，8-2，8-3

~B~CE -> 1-3，1-4，8-3，8-4

~B~D~E -> 1-1，2-1，7-1

~C~D~E -> 1-1，4-1，5-1，8-1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ABC  DE | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
| 00 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 01 | 1 |  |  |  | 1 | 1 | 1 | 1 |
| 11 | 1 |  |  |  | 1 | 1 | 1 | 1 |
| 10 | 1 |  |  |  | 1 | 1 | 1 | 1 |

# **时序逻辑作业**

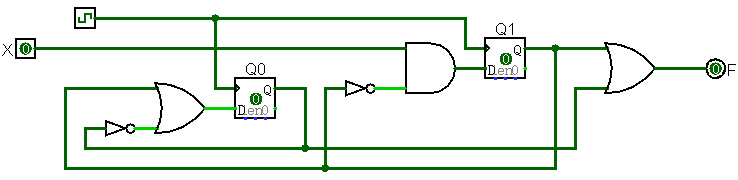
1. 某序列检测器，输入为data，输出为Q。当检测到1010时，Q输出1个周期的1，否则Q输出0。**给出状态机次态逻辑表达式和Q的逻辑表达式**。注意：{1010}为独立检测，不与后续序列拼接，即{101010}视为1次匹配成功。

答案：假设寄存器D2D1D0的编码000、001、010、011及100分别对应S0至S4。由于Q输出宽度以周期为单位，因此Q只能是状态的函数，即该状态机是摩尔型状态机。

**答案：**



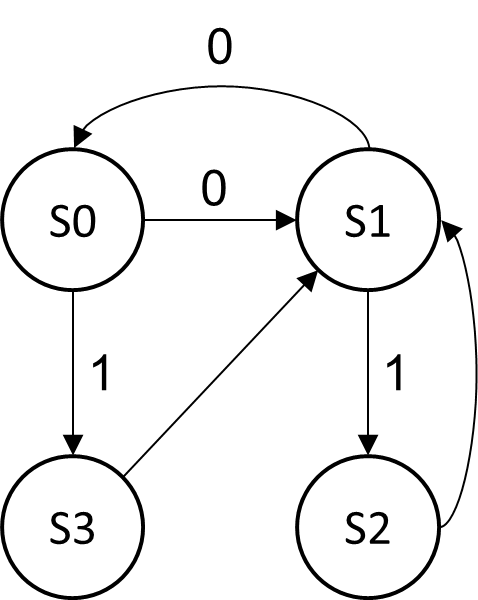
1. 状态机如下图所示。构造次态逻辑表达式和输出表达式，写出状态转换及输出表，画出状态图，分析状态机类型。



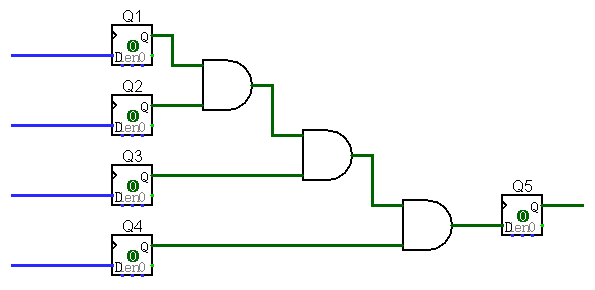
**答案：**

Moore型状态机，因为F只与状态寄存器相关。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Q1 | Q0 | X | Q1n | Q0n | F |  | 现态 | X | 次态 |
| 0 | 0 | 0 | 0 | 1 | 0 |  | S0 | 0 | S1 |
| 0 | 0 | 1 | 1 | 1 | 0 |  | S0 | 1 | S3 |
| 0 | 1 | 0 | 0 | 0 | 1 |  | S1 | 0 | S0 |
| 0 | 1 | 1 | 1 | 0 | 1 |  | S1 | 1 | S2 |
| 1 | 0 | 0 | 0 | 1 | 1 |  | S2 | 0 | S1 |
| 1 | 0 | 1 | 0 | 1 | 1 |  | S2 | 1 | S1 |
| 1 | 1 | 0 | 0 | 1 | 1 |  | S3 | 0 | S1 |
| 1 | 1 | 1 | 0 | 1 | 1 |  | S3 | 1 | S1 |

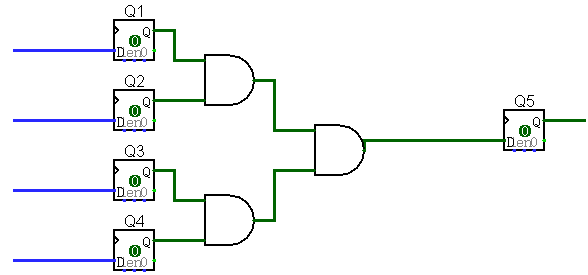


1. 对于如下电路，假设每个与门的延迟均为T，连线延迟、寄存器固有延迟等均为0。
   1. 计算关键路径的延迟以及寄存器的最大时钟频率。
   2. 重构电路以提高时钟频率，并计算提高比例。



**答案：**

1. 关键路径为3个AND门的串接，延迟为3T。最大时钟频率为1/3T。
2. 优化后为2层AND，延迟为2T。最大时钟频率为1/2T，比原设计频率提高50%。



1. 《数字设计和计算机体系结构》：第3.20题。

**答案：**

1. 由于输出与输入相关，因此这是一个Meely型状态机。
2. 2个寄存器Q1和Q0，编码值分别为：S0=0b00；S1=0b01；S2=0b10
3. 根据状态图得到如下真值表。关键要点在于对S2转S0的分析（表中绿色部分。无论是否对于化简有用，都一定先补全！）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 现态  S1S0 | A | B | 次态  S1S0 | F |
| 00 | 0 | X | 00 | 0 |
| 00 | 1 | X | 01 | 0 |
| 01 | X | 0 | 00 | 0 |
| 01 | X | 1 | 10 | 0 |
| 10 | 1 | 1 | 10 | 1 |
| 10 | 0 | 0 | 00 | 0 |
| 10 | 0 | 1 | 00 | 0 |
| 10 | 1 | 0 | 00 | 0 |

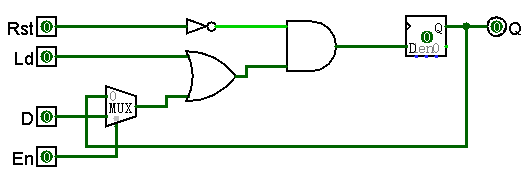
1. 表达式如下：
2. 功能：检测A和B先后输入为1，此后AB同时为1。
3. 用D触发器为核心，设计一个支持使能(En若为0，则寄存器值不变)、预置(Ld为1，则寄存器值为1)、清除(Rst为1，则寄存器被清除)功能的寄存器。寄存器输入信号为D，输出为Q。所有控制信号都是同步控制信号。3个信号的优先级从高到低为：Rst、Ld、En。

**答案：**

1. 因为3个控制信号都是同步控制信号，所以D触发器的输入端是3个控制信号及D的函数。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rst | Ld | En | D | Q | D触发器输入 |
| 1 | X | X | X | X | 0 |
| 0 | 1 | X | X | X | 1 |
| 0 | 0 | 0 | X | Q | Q |
| 0 | 0 | 1 | 0 | X | 0 |
| 0 | 0 | 1 | 1 | X | 1 |

1. 电路结构如下图



# **数据表示与运算方法作业**

1. 计算2223，2224，2225对应的十进制值。

**答案：**

2223 = 26；

2224 = 42；

2225 = 62

1. 分别计算0b1011101与0xB23的十进制值（均按无符号数处理）

**答案：**

0b1011101 = 1+4+8+16+64 = 93；

0xB23 = 0b1011\_0010\_0011 = 1+2+32+256+512+2048 = 2851

1. 用16进制方式表示无符号数10011111102。

**答案：**

0x27E

1. 除了十进制转换为二进制的方法，下面给出了从十进制转换为五进制和九进制的2个具体案例。根据该案例总结出从10进制转换为N进制的一般性方法。

表0‑1 十进制2007转换为5进制的计算过程

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 步骤 | 被除数 | 商 | 余数 | 位序 | 备注 |
| 1 | 2007 | 401 | 2 | 0 | 2007除以5 |
| 2 | 401 | 80 | 1 | 1 | 401除以5 |
| 3 | 80 | 16 | 0 | 2 | 80除以5 |
| 4 | 16 | 3 | 1 | 3 | 16除以5 |
| 5 | 3 | 0 | 3 | 4 | 被除数小于除数5，计算结束 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 4 | 3 | 2 | 1 | 0 | 位序 |
| 200710 | = | 3 | 1 | 0 | 1 | 2 | 5 |

表0‑2 十进制2018转换为9进制的计算过程

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 步骤 | 被除数 | 商 | 余数 | 位序 | 备注 |
| 1 | 2018 | 224 | 2 | 0 | 2018除以9 |
| 2 | 224 | 24 | 8 | 1 | 224除以9 |
| 3 | 24 | 2 | 6 | 2 | 24除以9 |
| 4 | 2 | 0 | 2 | 3 | 被除数小于除数9，计算结束 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | 3 | 2 | 1 | 0 | 位序 |
| 201810 | = | 2 | 6 | 8 | 2 | 9 |

**答案：**

组织循环：被除数对除数进行除法直至商为0；被除数调整为上一次的商。将所有的余数倒序排列就是结果。

1. 给出快速估算228的方法。

**答案：**

1. 给出6位二进制补码的表示范围。

**答案：**

{-32，+31}

1. 将下列十进制数转换为6位二进制补码并完成计算，同时指出结果是否存在溢出。

①16+15 ② 16+18 ③16-8 ④-16-16 ⑤-24-13

**答案：**

1）011111；

2）溢出；

3）001000；

4）100000；

5）溢出

1. 下列代码执行结束后，请用32位二进制补码方式分别表示c、s与us的值。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | char c ;  short s ;  unsigned short us ;  c = -1 ;  s = c ;  us = (unsigned short)c ; |

**答案：**

c：0xFFFF\_FFFF；

s：0xFFFF\_FFFF；

us=0x0000\_FFFF

# **指令和汇编作业**

1. 请从指令编码格式的角度分析MIPS寄存器个数为32个的合理性。提示：主要分析如果寄存器个数多于32时，对于指令编码会带来哪些负面影响。

**答案：**

多于32个，则编码位数就至少为6位。

* 1. R型移位指令：移位位数为4位，无法单条指令实现在32位内任意移位。
  2. I型计算：立即数部分则会减至14位。扩展至32位时至少需要2条指令。
  3. I型beq：转移地址范围大幅度减少。

1. 请以指令为单位分析jal的被调用函数的入口地址的范围。

**答案：**

如果将4G划分为16个256MB，则可以在jal所在的那个256MB区段内任意跳转。

1. 请结合C语言的特点，分析beq指令格式中imm域的合理性。

**答案：**

beq的imm有16位，相当于64K条指令的范围。意味着以beq为基准，其上下可跳转的范围为{-32K，+32K}。一条C语句大于在10条指令左右。这意味着if-else语句块的大下为6.4K条C语句。从程序设计合理性角度，不应该有这么大的语句块。因此imm的范围足够了。

1. 请把下列C代码翻译为汇编代码。假设$s0存储着变量i，可用的寄存器只有$s0~$s3。

do {

循环体；

while ( 0<i && i<100 ) ;

**答案：**

Loop :

循环体

blez $s0, Loop\_End // i ≤ 0 时退出

slt $s0, $s1, 100 // i ≥ 0 时, $s1为0

beq $s1, $0, Loop\_End //

j Loop

Loop\_End :

1. 请用beq和bne以及与分支无关的指令来完成下面这条语句的功能。提示：要防止溢出。

slt $s0, $s1, $s2

**答案：**

s1小于s2：s0为1

s1等于s2：s0为0

s1大于s2：s0为0

sub $t0, $s1, $s2

beq $t0, $0, SET0 // s1等于s2

srl $t0, $t0, 31 // 只保留符号位

beq $t0, $0, SET0 // 符号位为0：S1>S2

addi $s0, $0, 1 // 符号位为1：S1<S2

j END

SET0:

addi $s0, $0, 0

END:

1. 程序员编写了如下汇编程序来完成1000个字节的复制任务。假设循环开始前，$s0和$s1分别指向源字符串首地址和目的字符串首地址。

|  |  |
| --- | --- |
| LOOP: | lb $t0, 0($s0)  beq $t0, $0, TAIL  sb $t0, 0($s1)  addi $s0, $s0, 1  addi $s1, $s1, 1  j LOOP |
| TAIL: |  |

1. 请计算上述代码执行的指令总数。

**答案：**

1000字节，还得增加1个字节的结束符（即0），因此总计循环1001次。所以执行的总指令数=6\*1001=6006

1. 请最大化优化上述代码。优化代码只能使用本章讲授的指令。优化后代码仍然采用循环结构，且每次循环只能有1次复制操作。

**答案：**

循环250次用于复制，然后S1就指向了结束符所在的【字】。为此，需要写入字内的最高字节（因为最高字节是结束符）

addi $t1, $0, 250

Loop:

lw $t0, 0($s0)

sw $t0, 0($s1)

addi $s0, $s0, 4

addi $s0, $s1, 4

addi $t1, $t1, -1

bne $t1, 0, LOOP

sb $0, -3(s1)

1. 请计算优化后代码执行的指令总数。

**答案：**

指令总数=6\*250+1=1501

1. 请编写一个名为fib(n)的递归函数来计算第n个斐波那契数。斐波那契数列的计算公式为：f(n+2)=f(n+1)+f(n)，f(1)=f(2)=1。
2. 用MARS模拟器测试程序是否正确。

**答案：**

.data

vars: .space 0

.align 2

.align 1

str0: .asciiz "%d "

str1: .asciiz "Enter total number of terms: "

str2: .asciiz "Fibonacii series is : "

str3: .asciiz "%d\\t%d\\t"

str4: .asciiz [\\n](file:///\\n)

.text

main :

addi $sp, $sp, -8

addi $sp, $sp, -20

addi $t3, $0, 0

move $s0, $t3

sw $s0, 8($sp)

addi $t0, $0, 1

move $s1, $t0

sw $s1, 12($sp)

la $a0, str1

li $v0, 4

syscall

li $a0, '\n'

li $v0, 11

syscall

li $v0, 5

syscall

sw $v0, 20($sp)

la $a0, str2

li $v0, 4

syscall

li $a0, '\n'

li $v0, 11

syscall

la $a0, str3

li $v0, 4

syscall

li $a0, '\n'

li $v0, 11

syscall

lw $a0, 8($sp)

li $v0, 1

syscall

li $a0, '\n'

li $v0, 11

syscall

lw $a0, 12($sp)

li $v0, 1

syscall

li $a0, '\n'

li $v0, 11

syscall

lw $t0, 8($sp)

move $s0, $t0

lw $t1, 12($sp)

move $s1, $t1

lw $t3, 20($sp)

addi $t4, $0, 2

sub $t2, $t3, $t4

move $s2, $t2

jal LABEL1

la $a0, str4

li $v0, 4

syscall

li $a0, '\n'

li $v0, 11

syscall

addi $sp, $sp, 8

j exit

LABEL1:

addi $sp, $sp, -8

addi $sp, $sp, -4

addi $sp, $sp, -12

sw $s2, 16($sp)

sw $s1, 12($sp)

sw $s0, 8($sp)

sw $ra, 4($sp)

lw $t0, 16($sp)

addi $t1, $0, 0

ble $t0, $t1, LABEL3

lw $t3, 8($sp)

lw $t4, 12($sp)

add $t2, $t3, $t4

move $s0, $t2

sw $s0, 20($sp)

la $a0, str0

li $v0, 4

syscall

lw $a0, 20($sp)

li $v0, 1

syscall

li $a0, '\n'

li $v0, 11

syscall

lw $t0, 12($sp)

move $s0, $t0

sw $s0, 8($sp)

lw $t0, 20($sp)

move $s1, $t0

sw $s1, 12($sp)

lw $t0, 8($sp)

move $s0, $t0

lw $t1, 12($sp)

move $s1, $t1

lw $t3, 16($sp)

addi $t4, $0, 1

sub $t2, $t3, $t4

move $s2, $t2

jal LABEL1

b LABEL3

LABEL3:

lw $s2, 16($sp)

lw $s1, 12($sp)

lw $s0, 8($sp)

lw $ra, 4($sp)

addi $sp, $sp, 4

addi $sp, $sp, 20

jr $ra

exit:

1. 如果栈空间容量为4KB，请估算递归调用次数的极限。

答案：

假设函数需要入栈的寄存器总数为n（含PC以及通用寄存器等），则栈容量需求为4n字节。调用次数的极限为4K/4n=1K/n。

# **CPU作业**

1. 为了将 DM 结果向 ALU 转发，本章采用如图 6-16(a)的思路：从最后一级流水线寄存器MEM/WB 向 ALU 转发。但这个设计在执行如下指令序列时必须暂停一个时钟周期。设计师认为采用如图 6-16(b)的思路就可以解决这个问题：从直接从 DM 向 ALU 转发。虽然图 6-16(b)可以解决上述问题，但却使得流水线时钟频率下降了，请分析具体原因（假设 IM 读出、RF 读出、ALU、DM 读出的延迟均为 L）。

lw $1, xxx add yyy, $1, zzz

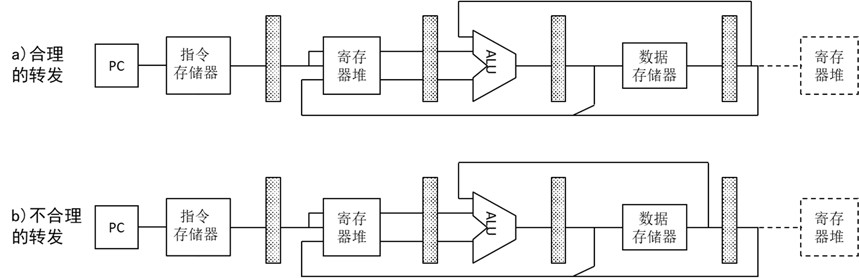


图 6-16 DM 转发的两种思路

**答案：**

由于从 DM 向 ALU 反馈，使得从 EX/MEM 寄存器到其自身之间存在了组合逻辑，即DM+ALU，其延迟为 2L。流水线频率变为 1/2L，因此图 b 流水线性能是图 a 的 50%。

1. 在流水中插入寄存器可以提高时钟频率。但是，随着级数增多，流水线性能提升会遇到瓶颈。首先，寄存器自身的时序开销（如寄存器建立时间与保持时间等）对于性能改善的影响越来越大。其次，随着流水线级数的增长，分支冒险会导致流水线排空的问题愈发严重。第三，数据冒险造成的暂停也会随之增多。

假设：5 级流水线 CPI 为 1.2，且每增加 1 级，CPI 增加 0.1；单周期 CPU 关键路径延迟为 800ps，寄存器自身时序开销为 50ps。

* 1. 建立 CPI 与流水线级数 N（N≥5）的计算公式。

**答案：**

* 1. 建立时钟周期延迟 Tc 与流水线级数 N 的计算公式。

**答案：**

* 1. 给出一条指令执行时间的计算公式。

**答案：**

一条指令执行时间

* 1. 请指出 N 为多少时，流水线性能最好。

**答案：**

N = 10，性能最好。

* 1. 请指出 N 为多少时，流水线性能改善最为显著。

**答案：**

N = 6，性能改善最为显著。

252

238

230

225

222

221

221

222

223

225

227

%

5

%

3

2

%

1

%

%

1

0

%

0

%

%

-1

%

-1

%

-1

%

-2

%

-1

0

%

1

%

%

2

%

3

4

%

5

%

6

%

205

210

215

220

225

230

235

240

245

250

255

5

7

8

9

10

11

12

13

14

15

6

指令延迟

)

ps

(

性能改善

)

%

(

1. 设计师将单周期数据通路改造为如图 6-17 所示的 3 级流水线。假设 RF 不支持内部转发。

ALU

寄存

器堆

数据

存储器

单周期

数据通路

流水线

数据通路

指令

存储器

寄存

器堆



PC

ALU

寄存

器堆

数据

存储器

指令

存储器

寄存

器堆

PC

取指令

译码

/

读操作数

/

执行

访存

/

回写

图 6-17 改造单周期数据通路为 3 级流水线

1. 流水线是否仍然可能会因为 beq 而需要清空流水线？如果会，最多有几条指令会被清除？

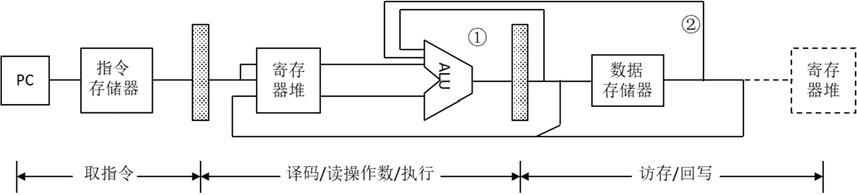
**答案：**

会。beq 的比较判断电路只能前移至译码/读操作数/执行阶段。这与本章介绍的处理是完全一致的。可能被清除的指令仍然是 beq 后面的那条指令，因此只有 1 条。

1. 假设指令集只有{lw，add}，请以 rs 寄存器为例，增加旁路以应对所有的数据冒险可能。给出思路即可，不用讨论因此带来的 MUX 及其控制。

**答案：**

从 ALU 和 DM 分别向 ALU 的 A 端转发数据。注意，由于 RF 无内部转发，因此必须有路径 2。



1. 对于第 2 问的指令集，能否消除 rs 寄存器相关的全部数据冒险？为什么？

**答案：**

可以。因为对于{lw，add}来说，rs 只在 ALU 的 A 端使用。无论当前哪条指令需要使用 ALU，其前面只可能有 1 条指令位于最后一级。而无论是哪条指令位于最后一级，结果均已产生，因此必然可以通过旁路转发来消除数据冒险。

1. 同样是从 DM 向 ALU 的转发，图 6-18(a)的设计会使得 5 级流水线性能下降。请分析图6-18(b)的设计会导致 3 级流水线性能下降吗？假设 IM 读出、RF 读出、ALU、DM 读出的延迟均为 L，忽略所有控制器延迟及 MUX 延迟。

ALU

寄存

器堆

数据

存储器

指令

存储器

寄存

器堆



PC

5级流水线：不合理的转发

ALU

寄存

器堆

数据

存储器

指令

存储器

寄存

器堆



PC

3级流水线：合理的转发

图 6-18 从 DM 向 ALU 的转发

**答案：**

1. 在 3 级流水线中，中间那段的延迟已经是最坏的 2L 了。
2. 转发电路的组合逻辑为 DM 读出+ALU，同样也是 2L。
3. 这表明转发并没有增加最坏延迟，因此不会导致性能下降。
4. 下图所示的流水线 CPU 执行如下指令序列。

I1 lw $1, 0($2)

I2 addi $1, $1, $1

I3 sw $1, 0($2) I4 lw $1, 4($2)

I5 sw $1, 8($2)

* 1. 分析上述指令执行过程中流水线共计需要暂停多少个时钟周期。

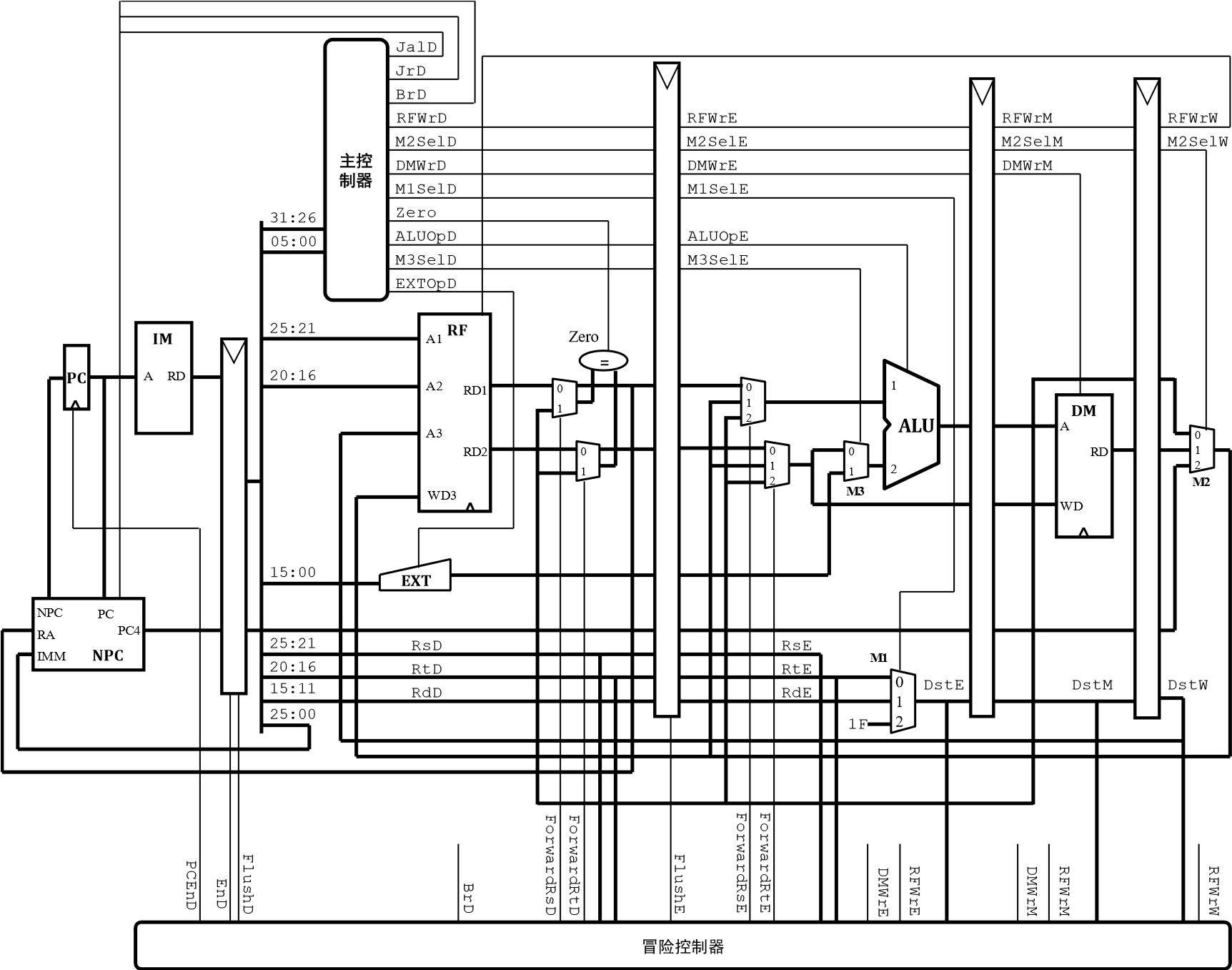
**答案：**

指令 1 和指令 2 之间在$1 有数据冒险，在转发电路存在的前提下，仍然必须暂停1 个时钟周期。指令序列调整为：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | lw $1, 0($2) nop  addi $1, $1, $1 sw $1, 0($2) lw $1, 4($2) sw $1, 8($2) |

原指令 4（现指令 5）与原指令 5（现指令 6）之间存在数据冒险。但原设计没有转发电路，因此必须将 sw 冻结在 IF/ID，直至 lw 进入 MEM/WB。此时通过 RF 内部转发，sw 就能得到正确的$1。暂停周期数为 2。 故上述代码总的暂停时间为 3 个时钟周期。

* 1. 是否可以增加转发来提升流水线性能？如果可以，请简述设计思路。



**答案：**

前问的 2 个暂停周期是因为解决 lw-sw 之间的数据冒险而产生的，因此需要从MEM/WB 增加一个转发至 DM 的 WD。

DM



WD

EX/MEM

MEM/WB

0

1

1. 在第 5 题图中所示的流水线 CPU 执行某程序，其指令分布如下：load 占 15%，store 为 10%，分支指令为 10%，R 型计算类指令为 65%。假设：load-R 导致暂停概率为 30%； load-store 导致的暂停概率为 5%；分支指令预测成功率为 75%。计算流水线执行该程序的 CPI。

**答案：**

1. load-R：没有数据数据相关时，load 的 CPI 为 1。如果有数据相关，需暂停 1 个时钟周期，其 CPI 为 2。
2. load-store：没有数据数据相关时，load 的 CPI 为 1。如果有数据相关，需暂停 2 个时钟周期，其 CPI 为 3。
3. store：题目中未出现数据相关，因此 CPIstore 为 1。
4. 分支：预测成功，分支的 CPI 为 1。如果预测失败，需暂停 1 个时钟周期，分支的CPI 为 2。
5. R 型：CPI 为 1。

1. 如图 6-19 所示，某 MIPS 标准 5 级流水线仅支持 M 级向 D 级的转发（注意：寄存器堆无内部转发）。某程序员编写了如下 MIPS 代码，请回答下列问题。

图

6-19

仅支持

M

级向

D

级转发的

5

级流水线

指令

存储器

A

RD

**PC**

寄存器堆

A1

A2

A3

WD

RD1

RD2

**ALU**

Zero

ALU

结果

数据

存储器

A RD

WD

25:21

20:16

15:11

**+**

4

0

1

E

M

W

D

扩展

15:00

0

1

0

1

0

1

I1: lw $1, 0($2)

I2: sw $1, 0($1)

I3: add $3, $2, $2

I4: sub $4, $1, $3

I5: or $5, $5, $6

* 1. 请指出上述指令片段在上述流水线中执行时存在的所有数据相关。

**答案：**

lw-sw 在$1 有 2 次相关；lw-sub 在$1 相关；add-sub 在$3 相关。

* 1. 请通过调整指令顺序来优化上述指令片段以最大化减少暂停。

**答案：**

I1: lw $1, 0($2)

I3: add $3, $2, $2

I5: or $5, $5, $6 I2: sw $1, 0($1)

I4: sub $4, $1, $3

* 1. 对于优化前和优化后指令片段，分别给出流水线的执行时间，并说明理由。示例：对于 2 条无冒险的指令片段，则流水线执行时间为 6 个时钟周期。

**答案：**

优化前：13 个 cycle

由于仅在 W 和 D 之间存在转发，因此 lw-sw 以及 add~sub 之间必须分别插入 2 个 NOP。指令总数从5条变为9条，因此流水线共计需要9+（5-1）=13个时钟周期。 优化后：9 个 cycle

优化后，lw~sw 和 add~sub 的数据相关均通过转发解决了，因此无需插入 NOP，故执行时间=5+（5-1）=9 个时钟周期

# **存储作业**

1. 一标准流水线 CPU 的理想 CPI 为 1，其执行的程序片段中 load 和 store 类指令占 30%，其余指令均为 R 型指令。主存延迟为 50 个时钟周期。L1 级 cache 性能为：命中时间为 1 个时钟周期，缺失率为 2%。L2 级 cache 性能为：命中时间为 10 个时钟周期，缺失率为 5%。针对如下情况分别计算流水线的实际 CPI：没有 cache；只有 L1 级 cache；L2 级 cache。

**答案：**

1. 由于没有 cache，因此指令和数据的缺失率均为 100%；

暂停周期数 = 指令缺失率 × 指令缺失代价 + 数据缺失率 × 数据缺失代价

= 50 + 30% × 50 = 65

1. 只有 L1 级 cache

暂停周期数 = 指令缺失率 × 指令缺失代价 + 数据缺失率 × 数据缺失代价

= 2% × 50 + 30% × 2% × 50

= 1 + 0.3

= 1.3

1. 有 L2 级 cache：

由于不分指令 cache 和数据 cache，因此 L1 指令缺失代价与 L1 数据缺失代价是相同的。

暂停周期数 = 指令缺失率 × 指令缺失代价 + 数据缺失率 × 数据缺失代价

= 2% × L1 缺失代价 + 30% × 2% × L1 缺失代价

= (2% + 0.6%) × (10 + 5% × 50)

= 0.026 × 12.5

= 0.325

1. 直接映射 cache 参数如下：cache 的数据容量为 16KB，cache 块为 16B。现该 cache 接入一个 32 位 cpu。请给出 cache 的 TIO 结构以及包含标记后的 cache 块容量。

**答案：**

1. Offset：log2(16)=4
2. Index：16KB/16=1024 块，log2(1024)=10
3. Tag：32-4-10=18
4. cache 块总容量：数据+Tag+Valid=16B+18b+1b=16B+19b
5. 组相联映射 cache 参数如下：cache 的数据容量为 512KB，16 路相联，cache 块为 32B。现该 cache 接入一个 32 位 cpu。请给出 cache 的 TIO 结构。

**答案：**

* 1. offset：log2(32)=5
  2. Index：512KB/32B=16K 块，16K/16=1K 组，log2(1K)=10
  3. Tag：32-5-10=17

1. 对于第 3 问的系统，cache 采用写回策略，cache 命中时间为 1 个时钟周期，从主存调入一块或回写一块的代价均为 100 个时钟周期。某程序片段将起始地址为 0000\_0000h 的 1MB 内存单元全部初始化为 0，代码如下所示。

int \*p=0x0 ;

for ( int i=0; i<1024\*1024/4; i++ )

\*p++ = 0 ;

* 1. 计算该程序片段中的数据访问缺失率（忽略指令缺失等因素）。

**答案：**

1. 1MB 主存对应的块数：1MB/32B=32K（块）
2. 由于是线性初始化，因此这 32K 块均会访问 1 次。可以看成是如下循环：
3. 第 1 次循环:

1. 当初始化主存块 0 字 0 时，缺失。于是从将主存块 0 写入 cache 的组 0 块 0。之后的 7 个字（8 字-1 字）均命中。

2. 当初始化主存块 1 字 0 时，缺失。于是从将主存块 1 写入 cache 的组 1 块0。之后的 7 个字（8 字-1 字）均命中。

3. 类似的，主存块 15，映射到 cache 组 15 块 0。

1. 第 2 次循环：

1. 主存块 16 字 0 时，缺失，会被映射到 cache 组 0 块 1；之后 7 个字均命中。

2. 依次类推，主存块 31 字 0 时，缺失，会被映射到 cache 组 15 块 1。之后 7 个字均命中。

1. 重复上述过程，直至第 1K 次循环。在第 1K 次循环中，主存块（16K-16）至主存块（16K-1）被分别装入 cache 组 0 块 15 和 cache 组 15 块 15。
2. 至此，主存的前 16K 块已经访问结束了，cache 所有块均被写满了，再无空闲块了。
3. vii. 而主存的后 16K 块的行为与前 16K 块完全一样，区别在于需要把前 16K 块写回主存。
4. 主存访问时，其对应的每个主存块的访问情况完全相同：字 0 缺失，字 1~字7 均命中。因此，数据访问缺失率=1/8=12.5%
   1. 计算该程序片段的数据访问实际需要多少个时钟周期。
5. 对于每个块来说，有 1 次缺失，其代价是 100 个时钟周期；7 次命中，7 个周期。故每个块是 100+7=107 个时钟周期。
6. 总共有 32K 块，因此访问时间=107\*32K（时钟周期）。
7. 但是后 16K 块还会导致 16K 次替换，每次替换都会有 100 时钟周期，因此替换时间=100\*16K（时钟周期）
8. 总时间=107\*32K+100\*16K=5024K（时钟周期）
   1. 计算程序片段的存储访问次数与实际时钟周期数之比。
9. 程序总存储访问次数=1MB/4=256K
10. 程序总存储访问次数：实际时钟周期数=256K：5024K≈1:20
11. 意味着每 20 个时钟周期才能完成一次存储访问。
12. 假设某系统的虚页和物理页尺寸均为 8KB，40 位虚地址，物理主存容量为 32GB。请问虚页号与物理页号的位数分别是多少。

**答案：**

* 1. 8KB 页面，则页内地址为 13 位。
  2. 虚页号（VPN）位数 = 40-13 = 27 位
  3. 物理页号（PPN）位数 = log2(32G)-13 = 35-13 = 22 位

1. 设计师给第 5 问的系统设计了 TLB。TLB 采用 2 路组相联结构，共有 256 个页表项。TLB中的每个页表项结构如下图所示。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Valid | Dirty | 访问权限 | TLB Tag | PPN |
| 1 位 | 1 位 | 2 位 |  |  |

1. 请问每个页表项的位数是多少。

**答案：**

TLB 的组数 = 256/2 = 128 组

TLB 的 index 位数 = log2(128) = 7

TLB 的 tag 位 = 虚地址位数-页内偏移位数-index 位数

= 40-13-7

= 20 位

页表项位数= 1+1+2+20+22 = **46 位**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Valid | Dirty | 访问权限 | TLB Tag | PPN |
| 1 位 | 1 位 | 2 位 | 20 | 22 |

1. 页表的总存储容量是多少位？

**答案：**

TLB总容量 = TLB每个页表项 x 表项数

= 46 x 256

= **11776位**

1. 操作系统设计团队希望将页面尺寸从 8KB 降低至 4KB，但硬件设计团队认为会增加硬件开销，因此不同意改变页面尺寸。你作为硬件设计师团队负责人，请陈述理由。

**答案：**

若页面尺寸改成4KB，页内偏移便会变成12位，虚页号变成28位，实页号变成23位。这样一来，页表项位数也会增大，TLB存储容量也会变大，所以硬件开销也自然会增大。

1. 现有 4K × 8 位容量的 DRAM 存储芯片。
   * 1. DRAM 芯片内置译码器输出的行/列选择线各是多少条。

**答案：**

DRAM 芯片共计 4K 个单元，因此内部共需

地址线的位数= log2(4K)=12，

即内部地址为 A[11:0]。由于 DRAM 采用行地址和列地址共享，因此行/列地址选择线均为12/2=6。

* + 1. DRAM 芯片内置的刷新地址计数器位数为多少位。

**答案：**

由于 DRAM 每次刷新一行，因此刷新地址计数器位数与行地址的位数相同，即刷新地址计数器位数为 6 位。

* + 1. DRAM 每刷新周期刷新一行存储单元是多少位。

**答案：**

一行的单元数由列地址数决定，即一行单元数=26=64 个。故总的存储单元位数为 64x8=512 位。

* + 1. 主存容量为 8K × 8 位，需使用 DRAM 芯片的数量。

**答案：**

需要的 DRAM 芯片数量 = (8K x 8）/ (4K x 8) = 2 片。

* + 1. 给出每个 DRAM 芯片的片选控制信号的逻辑表达式。

**答案：**

主存地址为 A[12:0]。每片 DRAM 芯片的位数为 A[11:0]，因此 A12 用来选择 2 个芯片。假设芯片 1 对应 0~4K，芯片 2 对应 4K-8K，则：

芯片 0 的片选=!A12

芯片 1 的片选= A12