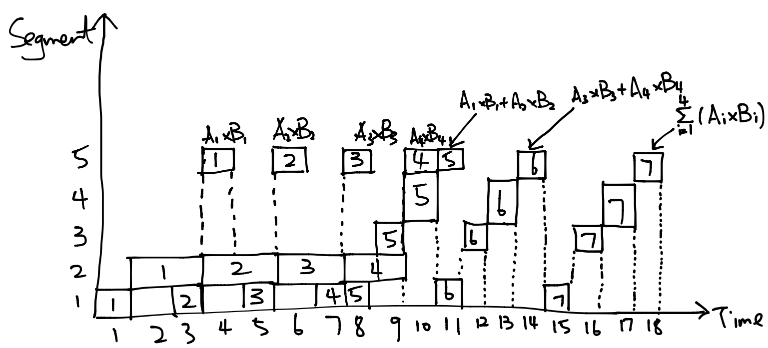
计算机系统结构第三次作业

李雨田　2010012193　计 14

April 5, 2014

3.8

如图所示, 可以先计算 $A_i \times B_i$, $i \in \{1,2,3\}$, 在计算 $A_4 \times B_4$ 之前先计算出 $A_1 \times B_1 + A_2 \times B_2$, 然后再算出剩下的值.



在 18 个 Δt 时间中, 给出了 7 个结果, 所以吞吐率为

$$TP = \frac{7}{18\Delta t}.$$

如果不适用流水线, 产生 7 个结果总共需要时间 $(4 \times 4 + 3 \times 4)\Delta t = 28\Delta t$, 所以加速比为

$$S = \frac{28\Delta t}{18\Delta t} = \frac{14}{9}.$$

流水线的效率可由阴影区的面积和总面积的比值求得

$$E = \frac{28}{5 \times 18} = \frac{14}{45}.$$

3.9

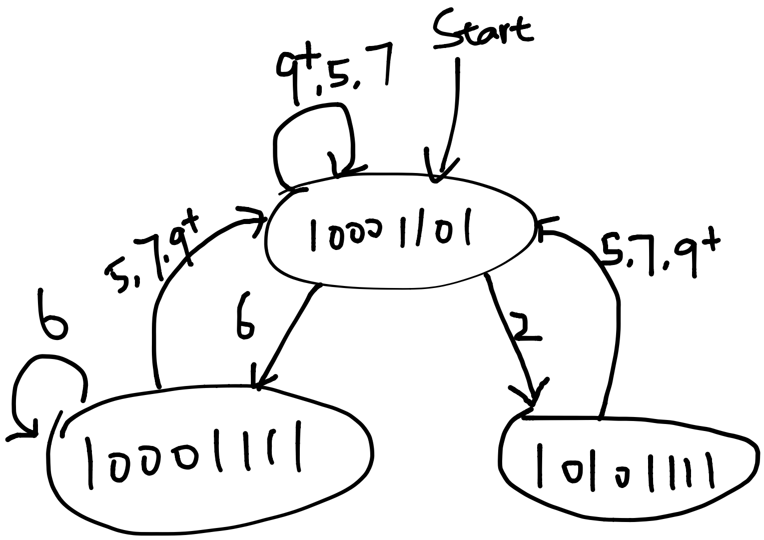
根据预约表, 可以得到禁止表

$$F = \{1, 3, 4, 8\}.$$

写出初始冲突向量

$$C_0 = (10001101).$$

再根据初始冲突向量可以画出状态转换图.



可以看出 $(2, 5)$ 是最优的调度策略, 平均时间间隔是 $3.5\Delta t$, 即可得吞吐率

$$TP = \frac{1}{3.5\Delta t} = \frac{2}{7\Delta t}.$$

如果连续输出 6 个任务, 分别相隔 $2\Delta t, 5\Delta t, 2\Delta t, 5\Delta t, 2\Delta t, 5\Delta t$ 进入流水线, 最后一个任务执行还需要时间 $9\Delta t$, 总共时间为 $30\Delta t$. 实际吞吐率为

$$TP = \frac{6}{30\Delta t} = \frac{1}{5\Delta t}.$$

实际吞吐率总是小于理论上的最大吞吐率, 这个结论得到验证.

3.10

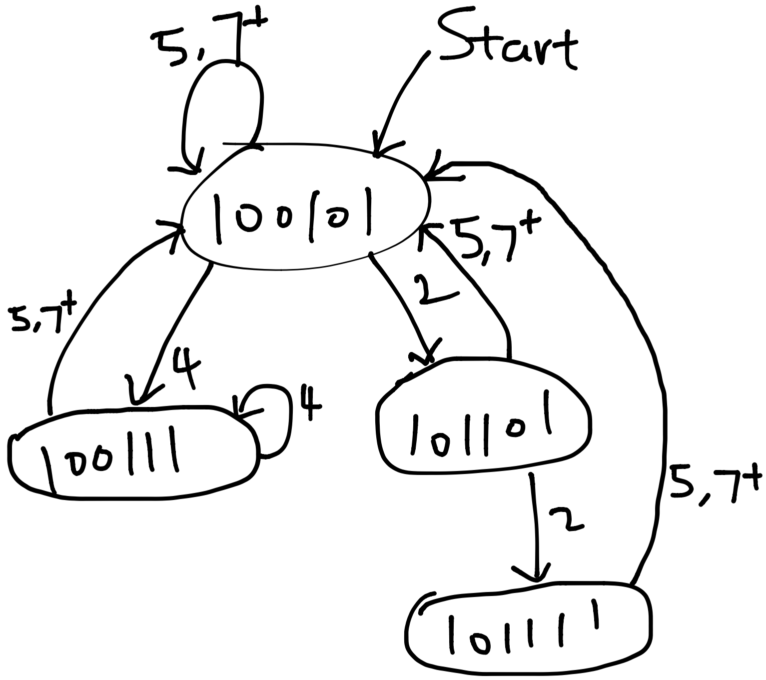
使用相同的流程, 首先根据预约表得到禁止表

$$F = \{1, 3, 6\}.$$

写出初始冲突向量

$$C_0 = (100101).$$

再根据初始冲突向量可以画出状态转换图.



可以看出允许不等时间间隔调度时,(2, 2, 5) 是最优的调度策略, 平均时间间隔是 $3\Delta t$, 可得到吞吐率

$$TF = \frac{1}{3\Delta t}.$$

等时间间隔调度时, 最优调度策略是 (5), 吞吐率

$$TF = \frac{1}{5\Delta t}.$$

连续输入 10 个任务时, 采用不等时间间隔调度耗时 $(2+2+5+2+2+5+2+2+5+2+7)\Delta t = 36\Delta t$, 实际吞吐率

$$TF = \frac{10}{36\Delta t} = \frac{5}{18\Delta t},$$

加速比为

$$S = \frac{10 \times 7\Delta t}{36\Delta t} = \frac{35}{18}.$$

采用等时间间隔调度则耗时 $(10 \times 5 + 7)\Delta t = 57\Delta t$, 实际吞吐率

$$TF = \frac{10}{57\Delta t},$$

加速比为

$$S = \frac{10 \times 7\Delta t}{57\Delta t} = \frac{70}{57}.$$

3.11

如果不使用任何其他定向硬件, 可以画出流水线时空图如下. 进行一次循环需要 18 个时钟周期, 并且两次循环只有 1 个时钟周期的重叠. 根据程序源代码不难算出总共需要循环 99 次, 所以整体执行需要 $98 \times (18 - 1) + 18 = 1684$ 个时钟周期.

这里用 I 表示 IF, D 表示 ID, E 表示 EX, M 表示 MEM, W 表示 WB, s 表示 stall.

LW	I	D	E	M	W													
DADDIU R1		I	s	s	D	E	M	W										
SW					I	s	s	D	E	M	W							
DADDIU R2								I	D	E	M	W						
DSUB									I	s	s	D	E	M	W			
BNEZ												I	s	s	D	E	M	W
LW2															I	s	s	I

如果有正常的定向路径, 可以画出流水线时空图如下. 这里采用的是预测分支失败的策略, 但是每次循环都是分支成功, 所以在执行 BNEZ 后的指令时, 第一次 IF 得到的指令会失败, 需要重新 IF 得到正确的指令. 这里进行一次循环需要 11 个时钟周期, 两次循环有 3 个周期的重叠. 整体需要 $98 \times (11 - 3) + 11 = 795$ 个时钟周期.

LW	I	D	E	M	W						
DADDIU R1		I	D	s	E	M	W				
SW				I	D	E	M	W			
DADDIU R2					I	D	E	M	W		
DSUB						I	D	E	M	W	
BNEZ							I	D	E	M	W
???								I	I	D	E

如果有一个单周期延迟分支. 注意到现在仅有的 stall 来自 LW 的延迟, 所以可以把 LW 指令放在 BNEZ 之前进行执行, 把 DADDIU R1 放在 BNEZ 之后的延迟槽进行执行. 由于循环的时候分支总是成功, 这样就把

LW 和 DADDIU R1 之间的一个时钟周期的延迟利用了起来, 延迟槽指令的结果也能每次都被采用. 流水时空图如下.

这样的话, 一次循环只要 10 个时钟周期, 两次循环有 4 个周期的重叠, 中间循环的 98 次共需要 $98 \times (10-4) = 588$ 个时钟周期. 但是第一次循环的 LW 和 DADDIU R1 之间的延迟是无法避免的, 多需要 3 个时钟周期, 最后一次循环要等流水线里的操作完全执行完, 还需要 9 个周期. 最后一次循环虽然 BNEZ 前仍然有 LW, 但是 BNEZ 会分支失败, 所以此 LW 指令在最后一步 WB 时要取消, 不能写回寄存器. 这个 LW 也是跟其他方式相比, 多执行的唯一一条指令. 这里总共需要 600 个时钟周期.

SW	I	D	E	M	W					
DADDIU R2		I	D	E	M	W				
DSUB			I	D	E	M	W			
LW				I	D	E	M	W		
BNEZ					I	D	E	M	W	
DADDIU R1						I	D	E	M	W
???							I	D	E	M