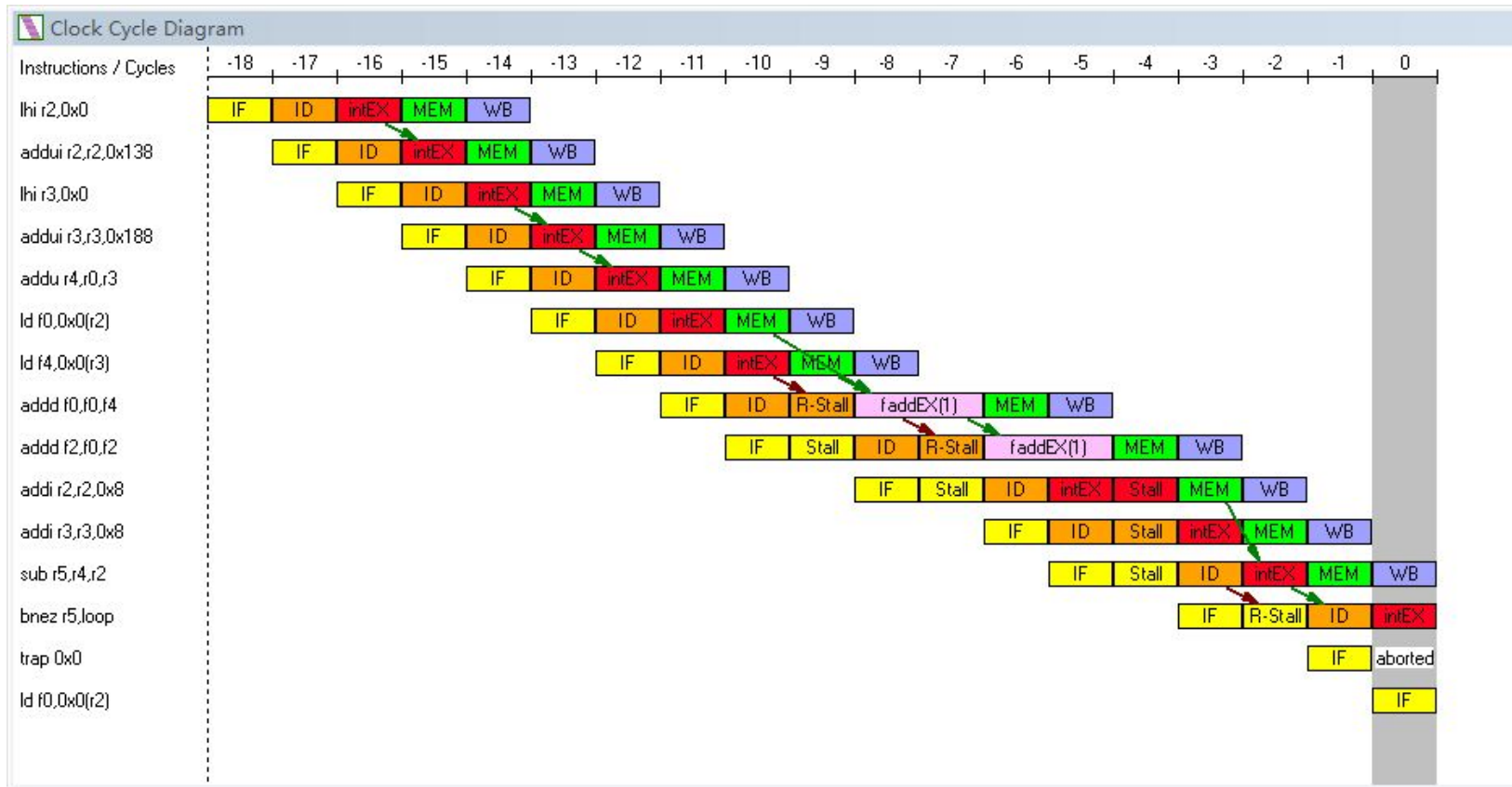
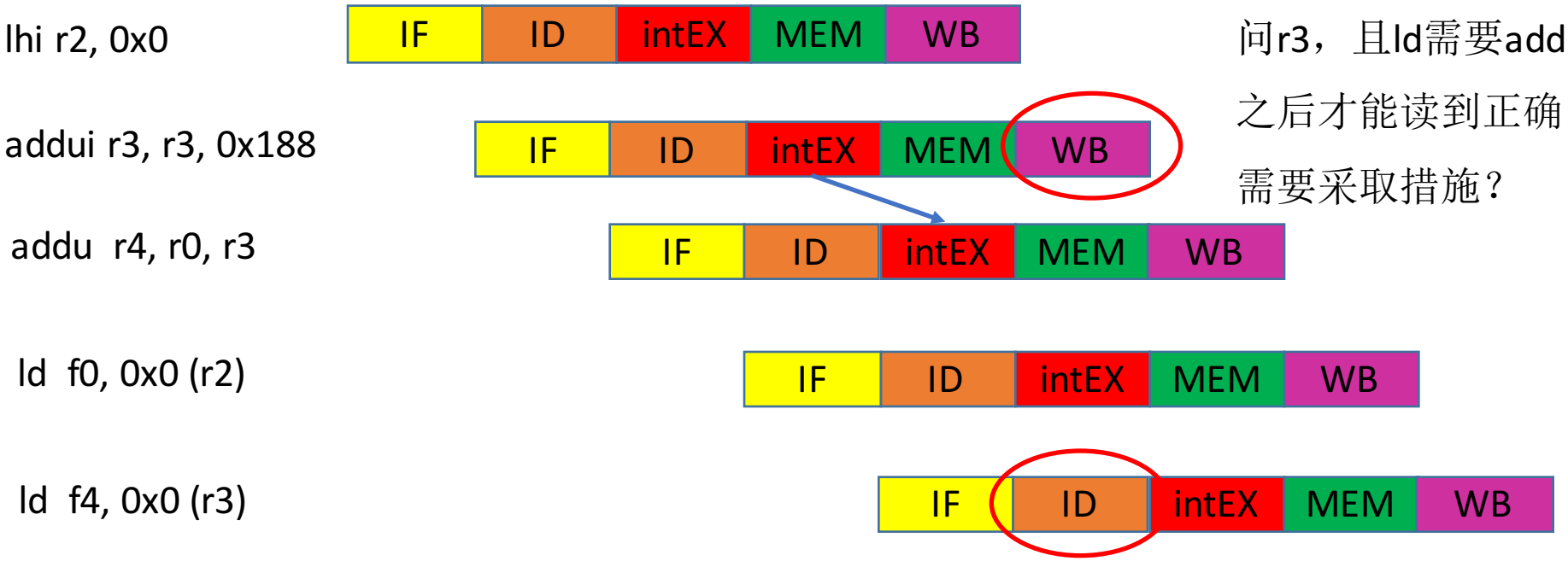
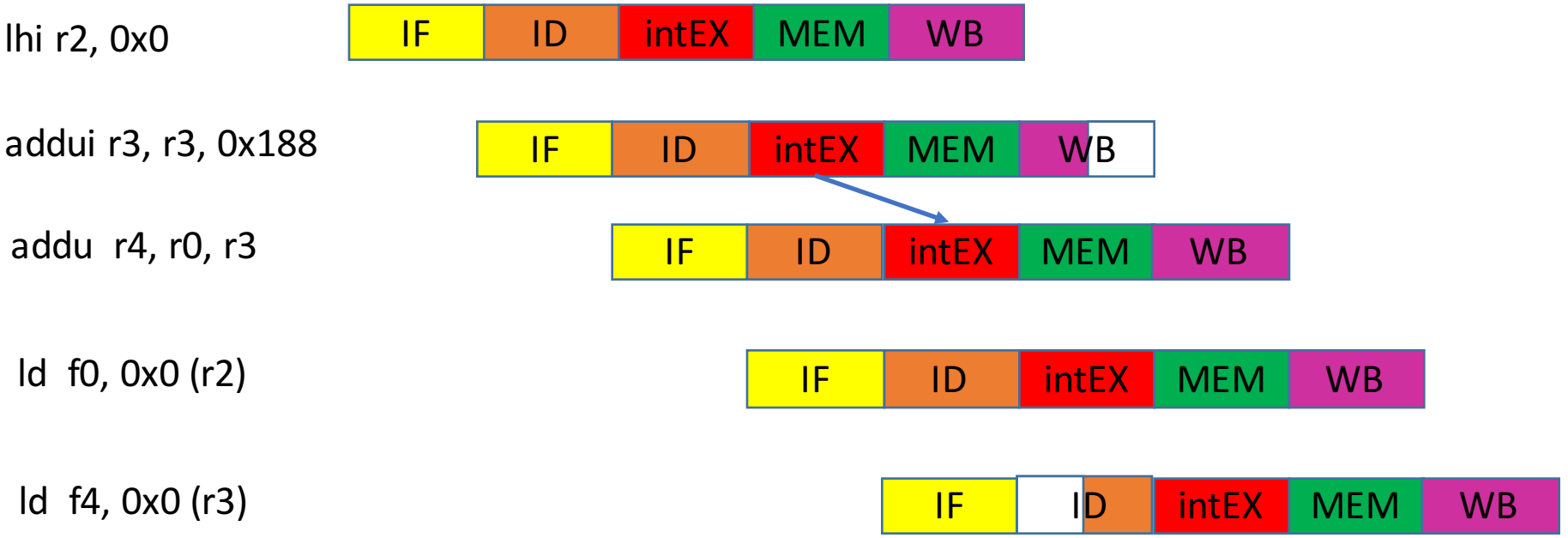
Hazards in Pipeline


Clock Cycle Diagram
Instructions / Cycles
-18 -17 -16 -15 -14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0
lhi r2,0x0
addui r2,r2,0x138
lhi r3,0x0
addui r3,r3,0x188
addu r4,r0,r3
ld f0,0x0(r2)
ld f4,0x0(r3)
addd f0,f0,f4
addd f2,f0,f2
addi r2,r2,0x8
addi r3,r3,0x8
sub r5,r4,r2
bnez r5,loop
trap 0x0
ld f0,0x0(r2)
IF
ID
intEX
MEM
WB
R-Stall
faddEX(1)
Stall
aborted

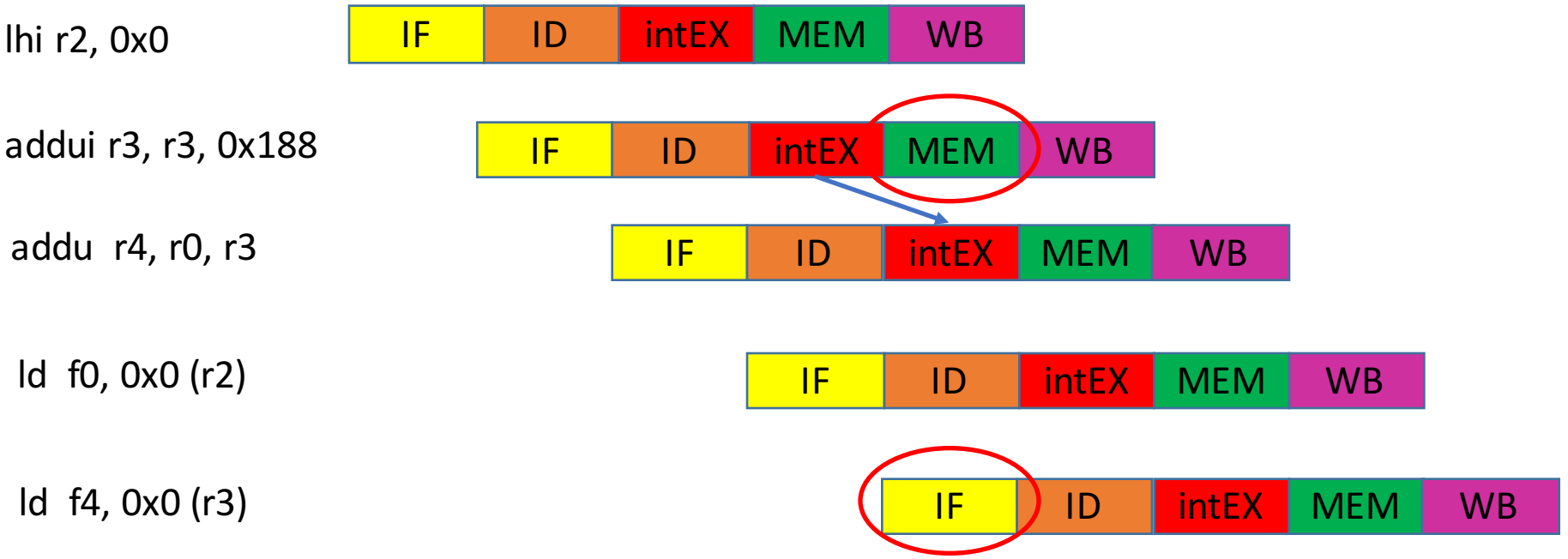
addui和ld之间存在相关，WB和ID都要访问r3，且ld需要addui将计算结果写回到r3之后才能读到正确的值，在这里为何不需要采取措施？



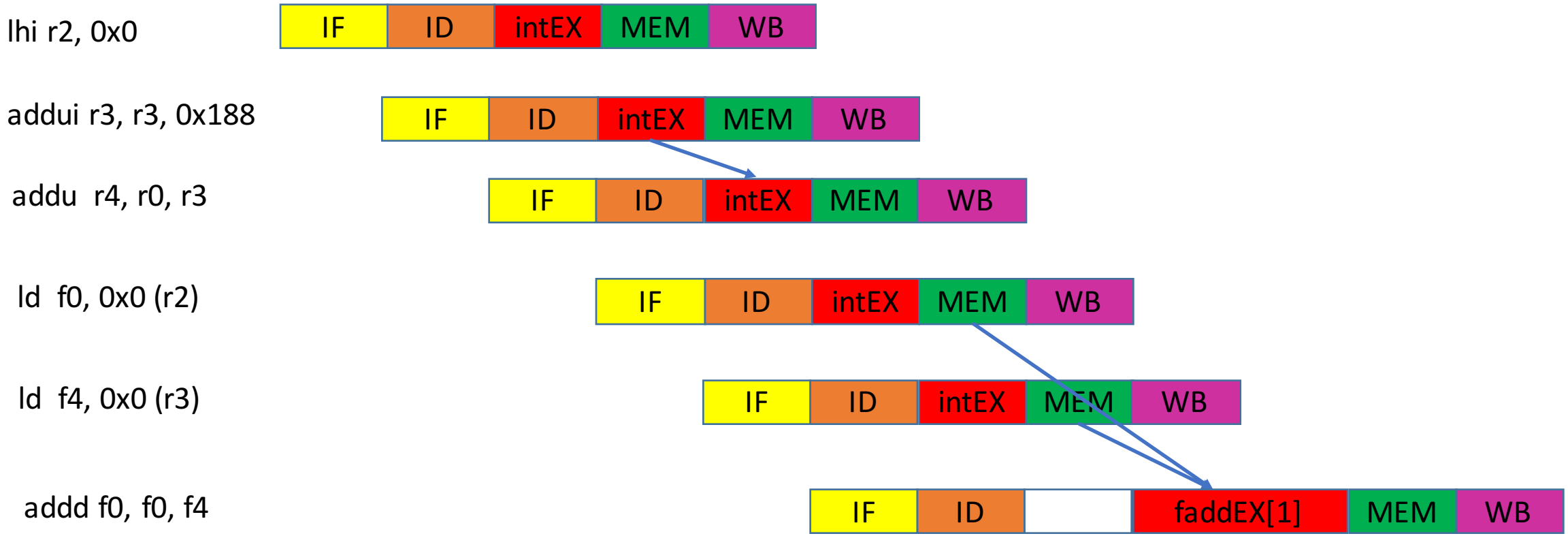
指令addui和addu发生数据相关，利用数据通路将addui的计算结果从它的intEX直接传送到addu的intEX



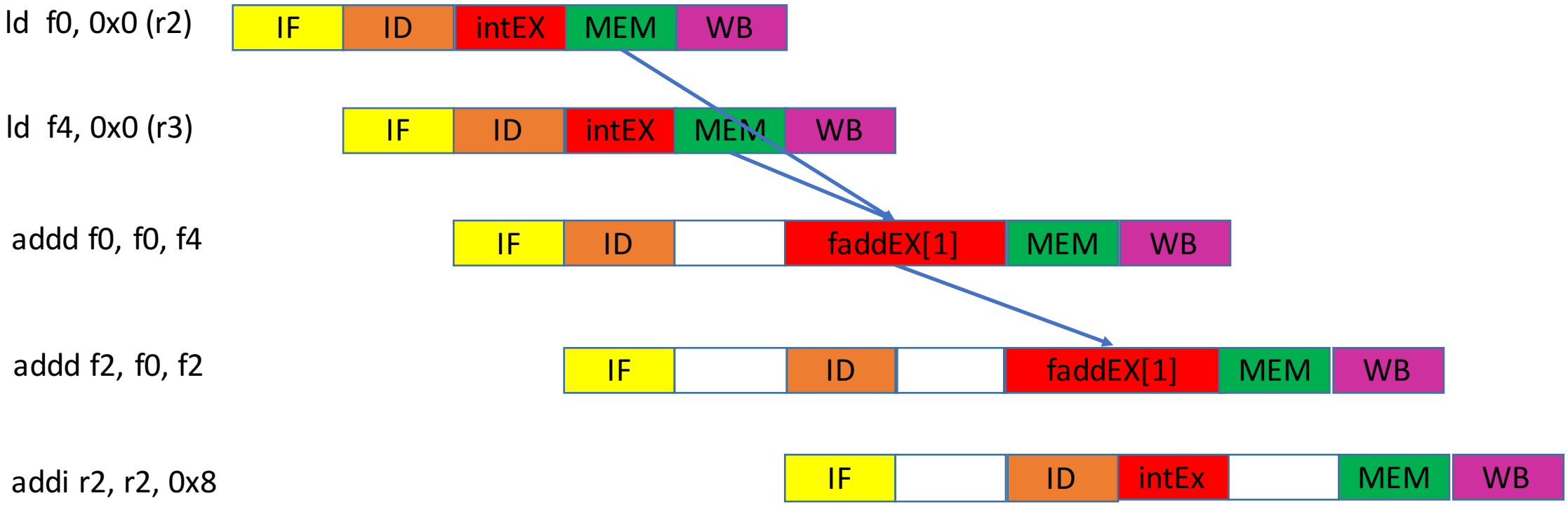
在WB的前半部分执行写操作，ID的后半部分进行读操作，所以不冲突



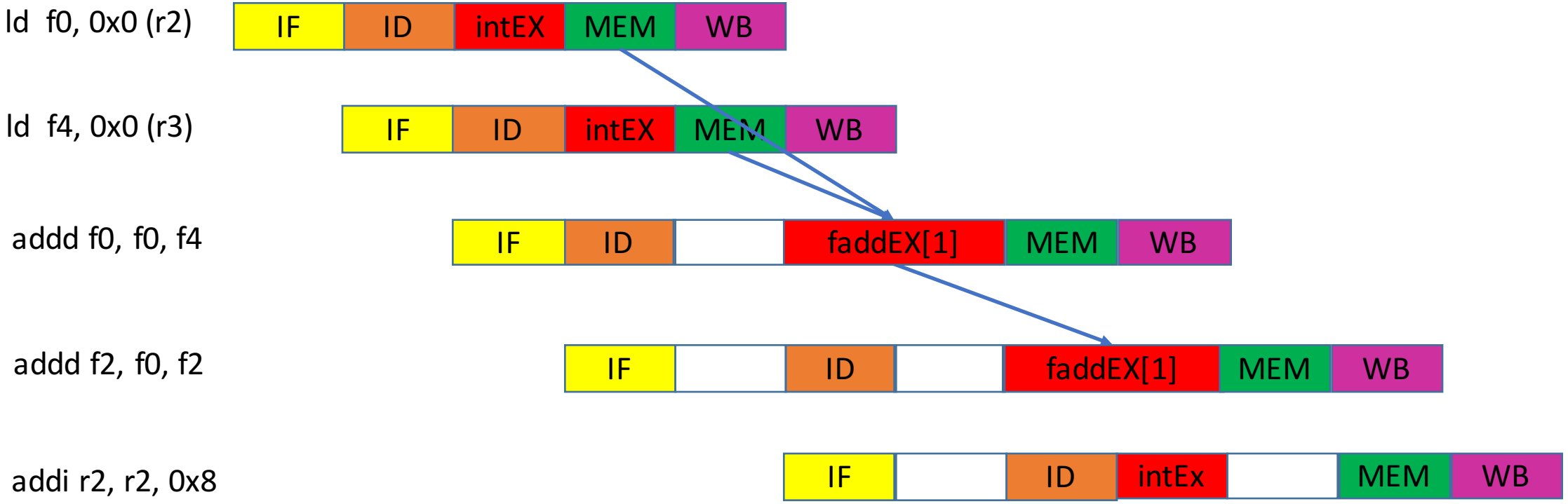
MEM和IF都要访问主存。可以让操作数和指令分别存在不同的cache，以实现MEM和IF的并行。另一种方法是采用指令预取，将近期可能访问的指令提前取到缓冲栈中。



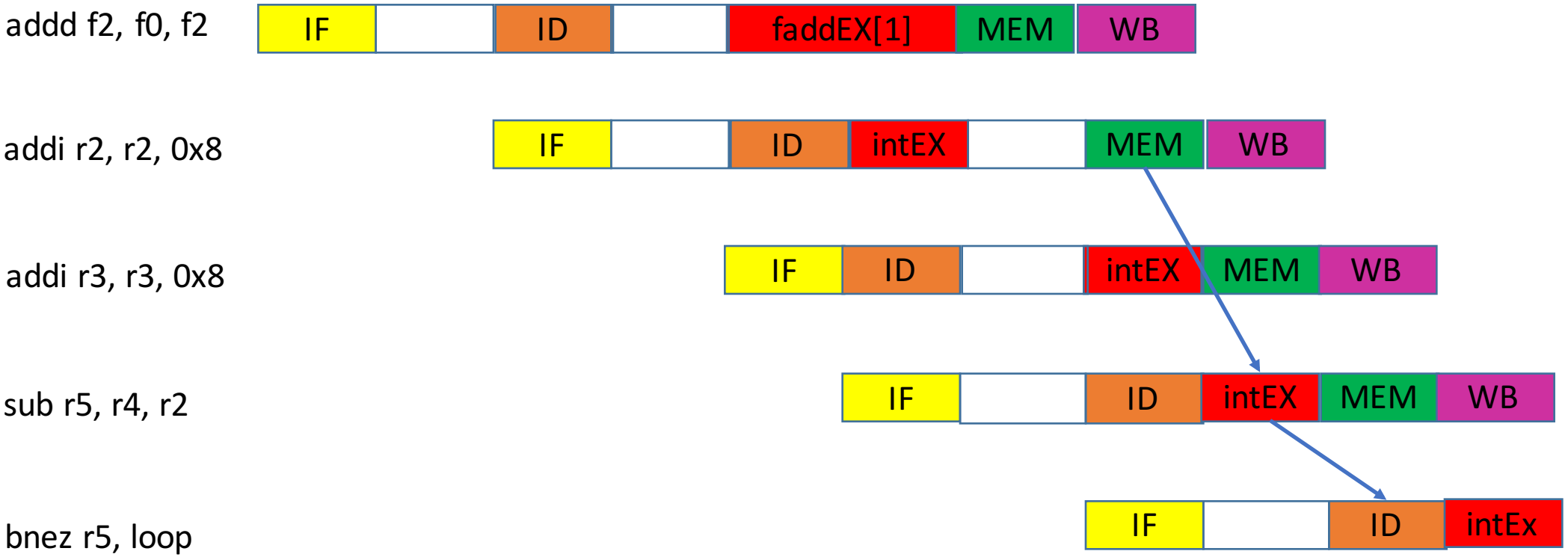
- 指令addd需要之前两条ld指令将数据从主存取出存储到f0和f4中之后才能取到正确的值
- 这里采用直接通路的方法，ld指令从主存中取到正确的操作数，经过专用数据通路将数据直接传送到addd的faddEX流水段
- 指令addd的faddEX流水段需要在ld f4, 0x0 (r3)访存之后才能进行，因此插入Stall



- 由于指令“addd f0, f0, f4”在ID之后是Stall，那么ID流水段的输出结果保存在ID与faddEX之间的缓冲寄存器中，若此时“addd f2, f0, f2”紧接执行ID，则破坏了该缓冲寄存器的内容，因此“addd f2, f0, f2”在IF之后需插入Stall
- “addd f2, f0, f2”与“addd f0, f0, f4”存在数据相关，需要设置专用通路将后者的计算结果直接输入到前者的faddEX流水段



- addd是浮点数加法，所需时间较长，“addd f2, f0, f2”需要等待“addd f0, f0, f4”流出faddEX之后才能使用该流水段
- 由于采用专用的浮点加法部件，所以faddEX和intEX可以做到并行


addd f2, f0, f2
IF
ID
faddEX[1]
MEM
WB
addi r2, r2, 0x8
IF
ID
intEX
MEM
WB
addi r3, r3, 0x8
IF
ID
intEX
MEM
WB
sub r5, r4, r2
IF
ID
intEX
MEM
WB
bnez r5, loop
IF
ID
intEx