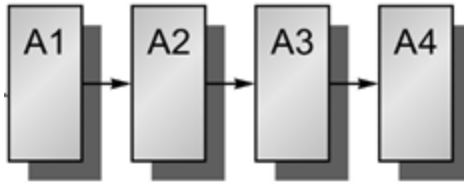
You can assume that



- All functional units are pipelined
- ALU operations take 1 cycle
- Memory operations take 3 cycles (includes time in ALU)
- Floating-point add instructions take 3 cycles
- Floating-point multiply instructions take 5 cycles
- There is no register renaming. No forwarding
- Instructions are fetched, decoded and issued in order
- The ISSUE stage is a buffer of limited length that holds instructions waiting to start execution
- An instruction will only enter the issue stage if it does not cause a WAR or WAW hazard
- Only one instruction can be issued at a time, and in the case multiple instructions are ready, the oldest one will go first
- Program Counter calculation for branches and jumps has been anticipated in the ISSUE stage

THIS MEANS THAT HAZARDS, IF THERE ARE ANY, DO NOT CAUSE STALLS IN F, D, IS STAGES

THOSE 2 INFORMATIONS ARE IMPLICITLY SAYING THAT THE ISSUE STAGE ACCEPTS ONLY 1 INSTRUCTION IN INPUT AT A TIME

OUTPUT

Code and Architecture

```
I1 lw.d $F3,B($R0)
I2 add.d $F2,$F2,$F3
I3 mul.d $F5,$F4,$F4
I4 addi $R0,$R0,8
I5 lw.d $F3,B($R0)
I6 add.d $F2,$F3,$F5
```

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

- IDENTIFY ALL THE CONFLICTS
- ANALYSE THE PROVIDED PIPELINES
- WRITE A CORRECT PIPELINE BASED ON THE ARCHITECTURE DESCRIBED AT POINT 3

```
I1 lw.d $F3,B($R0)
I2 add.d $F2,$F2,$F3
I3 mul.d $F5,$F4,$F4
I4 addi $R0,$R0,8
I5 lw.d $F3,B($R0)
I6 add.d $F2,$F3,$F5
```

RAW WAW

WAW RAW

RAW RAW

WAR

WAR

```
I1 lw.d $F3,B($R0)
I2 add.d $F2,$F2,$F3
I3 mul.d $F5,$F4,$F4
I4 addi $R0,$R0,8
I5 lw.d $F3,B($R0)
I6 add.d $F2,$F3,$F5
```

RAW $F3 I1-I2

WAW $F3 I1-I5

WAR $F3 I2-I5

RAW $F3 I5-I6

WAW $F2 I2-I6

WAR $F2 I2-I6

RAW $F5 I3-I6

WAR $R0 I1-I4

RAW $R0 I4-I5

Is this a possible execution? (1/4)

RAW I1-I2 $F3
WAR I1-I4 $R0
WAW I1-I5 $F3
WAR I2-I5 $F3

RAW I4-I5 $R0
RAW I3-I6 $F5
WAW I2-I6 $F2
RAW I5-I6 $F3

ALU OP: 1cc
MEM OP: 3cc
FP ADD: 3cc
FP MULT: 5cc

Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21
I1 lw.d $F3,B($R0)			F	D	IS	E1	E2	E3	W												
I2 add.d $F2,$F2,$F3				F	D	IS	E1	E2	E3	W											
I3 mul.d $F5,$F4,$F4					F	D	IS	E1	E2	E3	E4	E5	W								
I4 addi $R0,$R0,8	F	D	IS	E	W																
I5 lw.d $F3,B($R0)		F	D	IS	E1	E2	E3	W													
I6 add.d $F2,$F3,$F5							F	D	IS	E1	E2	E3	W								

Is this a possible execution? (1/4)

RAW I1-I2 $F3
WAR I1-I4 $R0
WAW I1-I5 $F3
WAR I2-I5 $F3

RAW I4-I5 $R0
RAW I3-I6 $F5
WAW I2-I6 $F2
RAW I5-I6 $F3

ALU OP: 1cc
MEM OP: 3cc
FP ADD: 3cc
FP MULT: 5cc

Fetch, Decode, Issue are not in order

Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21
I1 lw.d $F3,B($R0)			F	D	IS	E1	E2	E3	W												
I2 add.d $F2,$F2,$F3				F	D	IS	E1	E2	E3	W											
I3 mul.d $F5,$F4,$F4					F	D	IS	E1	E2	E3	E4	E5	W								
I4 addi $R0,$R0,8	F	D	IS	E	W																
I5 lw.d $F3,B($R0)		F	D	IS	E1	E2	E3	W													
I6 add.d $F2,$F3,$F5							F	D	IS	E1	E2	E3	W								

Is this a possible execution? (2/4)

RAW I1-I2 $F3
WAR I1-I4 $R0
WAW I1-I5 $F3
WAR I2-I5 $F3

RAW I4-I5 $R0
RAW I3-I6 $F5
WAW I2-I6 $F2
RAW I5-I6 $F3

ALU OP: 1cc
MEM OP: 3cc
FP ADD: 3cc
FP MULT: 5cc

Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21
I1 lw.d $F3,B($R0)	F	D	IS	E1	E2	E3	W														
I2 add.d $F2,$F2,$F3		F	D	IS	E1	E2	E3	W													
I3 mul.d $F5,$F4,$F4			F	D	IS	E1	E2	E3	E4	E5	W										
I4 addi $R0,$R0,8				F	D	IS	E	W													
I5 lw.d $F3,B($R0)					F	D	IS	E1	E2	E3	W										
I6 add.d $F2,$F3,$F5						F	D	IS	E1	E2	E3	W									

Is this a possible execution? (2/4)

RAW I1-I2 $F3
WAR I1-I4 $R0
WAW I1-I5 $F3
WAR I2-I5 $F3

RAW I4-I5 $R0
RAW I3-I6 $F5
WAW I2-I6 $F2
RAW I5-I6 $F3

ALU OP: 1cc
MEM OP: 3cc
FP ADD: 3cc
FP MULT: 5cc

Data hazards

MISSING STALLS WHERE NEEDED

Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21
I1 lw.d $F3,B($R0)	F	D	IS	E1	E2	E3	W														
I2 add.d $F2,$F2,$F3		F	D	IS	E1	E2	E3	W													
I3 mul.d $F5,$F4,$F4			F	D	IS	E1	E2	E3	E4	E5	W										
I4 addi $R0,$R0,8				F	D	IS	E	W													
I5 lw.d $F3,B($R0)					F	D	IS	E1	E2	E3	W										
I6 add.d $F2,$F3,$F5						F	D	IS	E1	E2	E3	W									

Is this a possible execution? (3/4)

RAW I1-I2 $F3
WAR I1-I4 $R0
WAW I1-I5 $F3
WAR I2-I5 $F3

RAW I4-I5 $R0
RAW I3-I6 $F5
WAW I2-I6 $F2
RAW I5-I6 $F3

ALU OP: 1cc
MEM OP: 3cc
FP ADD: 3cc
FP MULT: 5cc

Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23
I1 lw.d $F3,B($R0)	F	D	IS	E1	E2	E3	W																
I2 add.d $F2,$F2,$F3		F	D	IS	IS	IS	IS	E1	E2	E3	W												
I3 mul.d $F5,$F4,$F4			F	Ds	Ds	Ds	D	IS	E1	E2	E3	E4	E5	W									
I4 addi $R0,$R0,8				Fs	Fs	Fs	F	D	IS	E	W												
I5 lw.d $F3,B($R0)								F	Ds	Ds	D	IS	E1	E2	E3	W							
I6 add.d $F2,$F3,$F5									Fs	Fs	F	D	IS	IS	IS	IS	E1	E2	E3	W			

Is this a possible execution? (3/4)

RAW I1-I2 $F3
WAR I1-I4 $R0
WAW I1-I5 $F3
WAR I2-I5 $F3

RAW I4-I5 $R0
RAW I3-I6 $F5
WAW I2-I6 $F2
RAW I5-I6 $F3

ALU OP: 1cc
MEM OP: 3cc
FP ADD: 3cc
FP MULT: 5cc

Concurrent write ! SAME STAGE CANNOT SIMULTANEOUSLY EXECUTE FOR DIFFERENT INSTRUCTIONS

Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23
I1 lw.d $F3,B($R0)	F	D	IS	E1	E2	E3	W																
I2 add.d $F2,$F2,$F3		F	D	IS	IS	IS	IS	E1	E2	E3	W												
I3 mul.d $F5,$F4,$F4			F	Ds	Ds	Ds	D	IS	E1	E2	E3	E4	E5	W									
I4 addi $R0,$R0,8				Fs	Fs	Fs	F	D	IS	E	W												
I5 lw.d $F3,B($R0)								F	Ds	Ds	D	IS	E1	E2	E3	W							
I6 add.d $F2,$F3,$F5									Fs	Fs	F	D	ISs	ISs	ISs	IS	E1	E2	E3	W			

Is this a possible execution? (4/4)

RAW I1-I2 $F3
WAR I1-I4 $R0
WAW I1-I5 $F3
WAR I2-I5 $F3

RAW I4-I5 $R0
RAW I3-I6 $F5
WAW I2-I6 $F2
RAW I5-I6 $F3

ALU OP: 1cc
MEM OP: 3cc
FP ADD: 3cc
FP MULT: 5cc

Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23
I1 lw.d $F3,B($R0)	F	D	IS	E1	E2	E3	W																
I2 add.d $F2,$F2,$F3		F	D	IS	IS	IS	IS	E1	E2	E3	W												
I3 mul.d $F5,$F4,$F4			F	D	D	D	D	IS	E1	E2	E3	E4	E5	W									
I4 addi $R0,$R0,8				F	F	F	F	D	IS	E	E	W											
I5 lw.d $F3,B($R0)								F	D	D	IS	IS	E1	E2	E3	W							
I6 add.d $F2,$F3,$F5									F	F	D	D	IS	IS	IS	IS	E1	E2	E3	W			

Is this a possible execution? (4/4)

~~RAW I1-I2 $F3~~
~~WAR I1-I4 $R0~~
~~WAW I1-I5 $F3~~
~~WAR I2-I5 $F3~~

~~RAW I4-I5 $R0~~
~~RAW I3-I6 $F5~~
~~WAW I2-I6 $F2~~
~~RAW I5-I6 $F3~~

ALU OP: 1cc
MEM OP: 3cc
FP ADD: 3cc
FP MULT: 5cc

Correct execution

Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23
I1 lw.d $F3,B($R0)	F	D	I	E	E	E	W																
I2 add.d $F2,$F2,$F3		F	D	S	S	S	I	E	E	E	W												
I3 mul.d $F5,$F4,$F4			F	S	S	S	D	I	E	E	E	E	E	W									
I4 addi $R0,$R0,8							F	D	I	E	S	W											
I5 lw.d $F3,B($R0)								F	S	D	S	I	E	F	E	W							
I6 add.d $F2,$F3,$F5										F	D	S	S	S	S	I	E	E	E	W			

Is this a possible execution? (4/4)

RAW I1-I2 $F3
WAR I1-I4 $R0
WAW I1-I5 $F3
WAR I2-I5 $F3

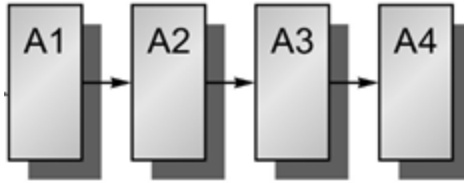
RAW I4-I5 $R0
RAW I3-I6 $F5
WAW I2-I6 $F2
RAW I5-I6 $F3

ALU OP: 1cc
MEM OP: 3cc
FP ADD: 3cc
FP MULT: 5cc

Correct execution

Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23
I1 lw.d $F3,B($R0)	F	D	IS	E1	E2	E3	W																
I2 add.d $F2,$F2,$F3		F	D	IS	IS	IS	IS	E1	E2	E3	W												
I3 mul.d $F5,$F4,$F4			F	D	D	D	D	IS	E1	E2	E3	E4	E5	W									
I4 addi $R0,$R0,8				F	F	F	F	D	IS	E	E	W											
I5 lw.d $F3,B($R0)								F	D	D	IS	IS	E1	E2	E3	W							
I6 add.d $F2,$F3,$F5									F	F	D	D	IS	IS	IS	IS	E1	E2	E3	W			

What if we change the buffer dimension?



- All functional units are pipelined
- ALU operations take 1 cycle
- Memory operations take 3 cycles (includes time in ALU)
- Floating-point add instructions take 3 cycles
- Floating-point multiply instructions take 5 cycles
- There is no register renaming. No forwarding
- Instructions are fetched, decoded and issued in order
- The issue stage is a buffer of unlimited length that holds instructions waiting to start execution
- An instruction will only enter the issue stage if it does not cause a WAR or WAW hazard
- Only one instruction can be issued at a time, and in the case multiple instructions are ready, the oldest one will go first
- Program Counter calculation for branches and jumps has been anticipated in the ISSUE stage

IN THIS CASE, MULTIPLE INSTRUCTIONS CAN SIMULTANEOUSLY LIVE IN THE ISSUE

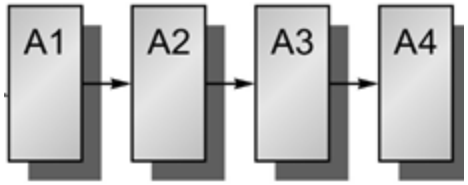
RAW I1-I2 $F3
WAR I1-I4 $R0
WAW I1-I5 $F3
WAR I2-I5 $F3

RAW I4-I5 $R0
RAW I3-I6 $F5
WAW I2-I6 $F2
RAW I5-I6 $F3

ALU OP: 1cc
MEM OP: 3cc
FP ADD: 3cc
FP MULT: 5cc

Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23
I1 lw.d $F3,B($R0)	F	D	IS	E1	E2	E3	W																
I2 add.d $F2,$F2,$F3		F	D	IS	IS	IS	IS	E1	E2	E3	W												
I3 mul.d $F5,$F4,$F4			F	D	^	^	^	IS	E1	E2	E3	E4	E5	W									
I4 addi $R0,$R0,8				F	F	F	F	D	IS	E	E	W											
I5 lw.d $F3,B($R0)								F	D	D	IS	IS	E1	E2	E3	W							
I6 add.d $F2,$F3,$F5									F	F	D	D	IS	IS	IS	IS	E1	E2	E3	W			

You can assume that



- All functional units are pipelined
- ALU operations take 1 cycle
- Memory operations take 3 cycles (includes time in ALU)
- Floating-point add instructions take 3 cycles
- Floating-point multiply instructions take 5 cycles
- There is no register renaming. No forwarding
- Instructions are fetched, decoded and issued in order
- The ISSUE stage is a buffer of unlimited length that holds instructions waiting to start execution
- An instruction will only enter the issue stage if it does not cause a WAR or WAW hazard
- Only one instruction can be issued at a time, and in the case multiple instructions are ready, the oldest one will go first
- Program Counter calculation for branches and jumps has been anticipated in the ISSUE stage
- The target address for a branch is available in the FETCH stage

Code and Architecture

Assembly Code:

I1: FOR: ld $f2, VB($r6)
I2: fadd $f3, $f2, $f6
I3: st $f3, VA($r7)
I4: ld $f3, VC($r6)
I5: st $f3, VC($r7)
I6: fadd $f4,$f4,$f3
I7: addi $r6, $r6, 4
I8: addi $r7, $r7, 4
I9: blt $r7, $r8, FOR

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

- ITEMIZE ALL CONFLICTS
- DRAW PIPELINE SCHEMA

```
I1: FOR: ld $f2, VB($r6)
I2: fadd $f3, $f2, $f6
I3: st $f3, VA($r7)
I4: ld $f3, VC($r6)
I5: st $f3, VC($r7)
I6: fadd $f4,$f4,$f3
I7: addi $r6, $r6, 4
I8: addi $r7, $r7, 4
I9: blt $r7, $r8, FOR
```

RAW $f2 I1-I2

RAW $f3 I2-I3

WAR $f3 I3-I4

WAW $f3 I2-I4

RAW $f3 I4-I5

RAW $f3 I4-I6

WAR $r6 I1-I7

WAR $r6 I4-I7

RAW $r7 I8-I9

WAR $r7 I3-I8

WAR $r7 I5-I8

CNTRL

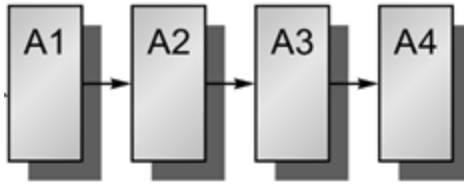
Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

I3 BNDS WRITE AT 1st/2 OF CC15 => RAW SOLVED
=> AT 2ND/2 OF CC, I CAN START

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	Notes
1	FOR: ld $f2,VB($r6)	F	D	I	E	E	E	W																						
2	fadd $f3, $f2, $f6		F	D	s	s	s	I	E	E	E	W																		RAW I1 - I2 F2
3	st $f3, VA($r7)			F	D	s	s	s	s	s	s	I	E	E	E	W														RAW I2 - I3 F3
4	ld $f3, VC($r6)				F	s	s	s	s	s	s	s	s	s	D	I	E	E	E	W										WAR I3-I4 F3 WAW I2-I4 F3
5	st $f3, VC($r7)														F	D	s	s	s	I	E	E	E	W						RAW I4 - I5 F3
6	fadd $f4,$f4,$f3															F	D	s	s	s	I	E	E	E	W					RAW I4 - I6 F3
7	addi $r6, $r6, 4																F	s	D	s	s	I	E E1	s E1	s E1	W				WAR I1-I7 r6 WAR I4-I7 r6,
8	addi $r7, $r7, 4																		F	s	s	s	D	s	I	E E1	W			WAR I3-I8 r7, WAR I5-I8 r7, STRUCT ALU
9	blt $r7, $r8, FOR																						F	D	s	s	I	E	W	RAW R7 I8-I9
10	(New Loop Iteration)																											F	D	CNTRL

You can assume that



- All functional units are pipelined
- ALU operations take 1 cycle
- Memory operations take 2 cycles (includes time in ALU)
- Floating-point add instructions take 2 cycles
- Floating-point multiply instructions take 3 cycles
- There is no register renaming. No forwarding
- Instructions are fetched, decoded and issued in order
- The ISSUE stage is a buffer of unlimited length that holds instructions waiting to start execution
- An instruction will only enter the issue stage if it does not cause a WAR or WAW hazard
- Only one instruction can be issued at a time, and in the case multiple instructions are ready, the oldest one will go first
- Program Counter calculation for branches and jumps has been anticipated in the ISSUE stage

Exe Complex Pipeline: the Code

```
LOOP:I1: LD F1, 0 (R2)
     I2: MULTD F2, F1, F1
     I3: ADDD F3, F1, F5
     I4: MULTD F2, F3, F1
     I5: SUBD F5, F1, F5
     I6: SUBI R2, R2, 4
     I7: BNEZ R2, LOOP
```

- ITEMIZE ALL CONFLICTS
- DRAW PIPELINE SCHEMA

```
LOOP:I1:  LD   F1, 0 (R2)
     I2:  MULTD F2, F1, F1
     I3:  ADDD  F3, F1, F5
     I4:  MULTD F2, F3, F1
     I5:  SUBD  F5, F1, F5
     I6:  SUBI  R2, R2, 4
     I7:  BNEZ  R2, LOOP
```

RAW F1 I1-I2
RAW F1 I1-I3
RAW F1 I1-I4
RAW F1 I1-I5

WAW F2 I2-I4

RAW F3 I3-I4

WAR F5 I3-I5

WAR R2 I1-I6

RAW R2 I6-I7

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 2 cycles
FP ADD: 2 cycles
FP MULT: 3 cycles

CC 0

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	Notes
1	LOOP: LD F1,0(R2)	F	D	I	E	E	W														
2	MULTD F2,F1,F1		F	D	S	S	I	E	E	E	W										RAW F1 I1-I2
3	ADDD F3,F1,F5			F	D	S	S	I	E	E	S	W									RAW F1 I1-I3
4	MULTD F2,F3,F1				F	S	S	S	S	D	S	I	E	E	E	W					RAW F1 I1-I4 RAW F3 I3-I4 WAW F2 I2-I4
5	SUBD F5,F1,F5									F	D	S	I	E	E	S	W				RAW F1 I1-I5 WAR F5 I3-I5
6	SUBI R2,R2,4										F	D	S	I	E	S	S	W			WAR R2 I1-I6
7	BNEZ R2, LOOP											F	D	S	S	S	S	I	E	W	RAW R2 I6-I7
8	(New Instruction)																		F	D	CNTRL