

Video explanation of solution is provided below the problem.

Beta Junkyard

4/4 points (ungraded)

You've discovered a secret room in the basement of the Stata center full of discarded 5-stage pipelined Betas. Unfortunately, many have certain defects. You discover that they fall into four categories:

- C1: Completely functional 5-stage Betas with working bypass paths, annulment, and other components.
- C2: Betas with a bad register file: all data read from the register file is zero.
- C3: Betas without bypass paths: all source operands come from the register file.
- C4: Betas without annulment of instructions following branches.

To help sort the Beta chips into the above classes, you write the following small test program:

```
. = 0x0
// Start at 0x0, with ZERO in all registers
ADDC(R31, 4, R0)
BEQ(R31, X, R2)
MULC(R2, 2, R2)
X: SUBC(R2, 4, R2)
  ADD(R0, R2, R3)
  JMP(R3)
```

Your plan is to single-step through the program using each Beta chip, carefully noting the address the final JMP loads into the PC. Your goal is to determine which of the above four classes a chip falls into by this JMP address.

For each class of Beta processor described above, specify the value that will be loaded into the PC by the final JMP instruction.

Pipeline diagram showing first 7 cycles of test program executing on C1:

Cycle	0	1	2	3	4	5	6
IF	ADDC	BEQ	MULC	SUBC	ADD	JMP	
RF		ADDC	BEQ	NOP	SUBC	ADD	JMP
ALU			ADDC	BEQ	NOP	SUBC	ADD

MEM

ADDC BEQ NOP SUBC

WB

ADDC BEQ NOP

C1: JMP goes to address:

✓ Answer: 8

C2: JMP goes to address:

✓ Answer: 4

C3: JMP goes to address:

✓ Answer: 0

C4: JMP goes to address:

✓ Answer: 16

Explanation

C1: Fully functioning pipelined beta with bypass paths.

ADDC(R31, 4, R0) – eventually write 4 to R0
 BEQ(R31, X, R2) – eventually write $PC + 4 = 8$ into R2
 skip MULC because branch was taken
 SUBC(R2, 4, R2) – gets the value of R2 from the bypass path, so it does $8 - 4 = 4$ and eventually writes that to R2
 ADD(R0, R2, R3) – reads R0 from the register file so it reads a 4 for R0, and reads R2 from the bypass path so it reads 4 for R2 – writes 8 into R3
 JMP(R3) – reads R3 from bypass path so it jumps to address 8

C2: Anytime you read something directly from the register file, you will read a 0, but if you are reading a value from a bypass path, then you will read the correct value.

ADDC(R31, 4, R0) – eventually write 4 to R0
 BEQ(R31, X, R2) – eventually write $PC + 4 = 8$ into R2
 skip MULC because branch was taken
 SUBC(R2, 4, R2) – gets the value of R2 from the bypass path, so it does $8 - 4 = 4$ and eventually writes that to R2
 ADD(R0, R2, R3) – reads R0 from the register file so it reads a 0 for R0, and reads R2 from the bypass path so it reads 4 for R2 – writes 4 into R3
 JMP(R3) – reads R3 from bypass path so it jumps to address 4

C3: There are no bypass paths so all register values are read from the register file even if they are stale.

ADDC(R31, 4, R0) – eventually write 4 to R0
BEQ(R31, X, R2) – eventually write $PC + 4 = 8$ into R2
skip MULC because branch was taken
SUBC(R2, 4, R2) – reads $R2 = 0$ from the register file and subtracts 4 so it eventually writes -4 into R2
ADD(R0, R2, R3) – reads $R0 = 4$ from the register file and reads old $R2 = 0$ from register file – so it eventually writes 4 into R3
JMP(R3) – jumps to address 0 using original value of R3. (so you actually don't even need to do all the steps above).

C4: This beta has no branch annulment.

ADDC(R31, 4, R0) – eventually write 4 to R0
BEQ(R31, X, R2) – eventually write $PC + 4 = 8$ into R2
MULC(R2, 2, R2) – executed because there is no branch annulment, eventually writes $8 * 2 = 16$ into R2.
SUBC(R2, 4, R2) – gets the value of R2 from the bypass path, so it does $16 - 4 = 12$ and eventually writes that to R2
ADD(R0, R2, R3) – reads R0 from the register file so it reads a 4 for R0, and reads R2 from the bypass path so it reads 12 for R2 – writes 16 into R3
JMP(R3) – reads R3 from bypass path so it jumps to address 16

Submit

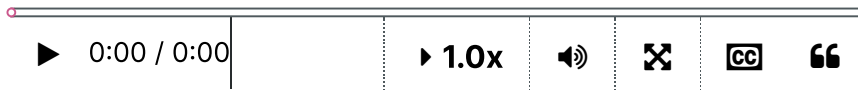
i Answers are displayed within the problem

Beta Junkyard

[Start of transcript. Skip to the end.](#)



For this problem, assume that you have discovered a




room full of discarded 5-stage pipelined betas.


These betas fall into four categories.


The first is completely functional 5-stage Betas with full bypass and annulment logic.

Video

 [Download video file](#)

Transcripts

 [Download SubRip \(.srt\) file](#)

 [Download Text \(.txt\) file](#)

Discussion

Hide Discussion

Topic: 15. Pipelining the Beta / WE15.2

Add a Post

Show all posts ▼

by recent activity ▼

There are no posts in this topic yet.

