Introduction to Architecture Exam

ExamTag: ALL-EXAMS

Instructions

Format: This exam consists of 6 multiple choice questions for each of the 12 lectures in the course. You may have to do some calculations in order to determine the correct answer. The motivation for having so many questions is to get a more accurate assessment of your knowledge of the material.

Be sure to circle the answers on the page titled "answers page" or your answers will not be graded. If you choose not to answer a question do not circle a letter. Feel free to detach the answers page from the remainder of the exam and only turn it in.

Grading: To discourage guessing, each incorrect answer will be worth -1/3 point, while each correct answer is worth +1 point. I.e., if you guess randomly, the expected score is 0.

Material: You are allowed a calculator and one double-sided, hand-written A4 sheet of notes during the exam.

Good luck!

Note: Please take this exam seriously and pass it! (Exams take huge amounts of teaching time to produce and grade, and most students and teachers would prefer that the time be spent teaching instead.)

Answers page.
Only answers circled here will be graded.

0. Optional anonymous background questions

0.1. A B C
0.2. A B C
0.3. A B C D
0.4. Program: _____

1. ISA 1

1.1. A B C D
1.2. A B C D
1.3. A B C D
1.4. A B C D
1.5. A B C D
1.6. A B C D

2. ISA 2

2.1. A B C D
2.2. A B C D
2.3. A B C D
2.4. A B C D
2.5. A B C D
2.6. A B C D

3. Computer Arithmetic

3.1. A B C D
3.2. A B C D
3.3. A B C D
3.4. A B C D
3.5. A B C D
3.6. A B C D

4. Logic

4.1. A B C D
4.2. A B C D
4.3. A B C D
4.4. A B C D
4.5. A B C D
4.6. A B C D

5. Processor Control and Datapath

5.1. A B C D
5.2. A B C D
5.3. A B C D
5.4. A B C D
5.5. A B C D
5.6. A B C D

6. Pipelining

6.1. A B C D
6.2. A B C D
6.3. A B C D
6.4. A B C D
6.5. A B C D
6.6. A B C D

7. Hazards

7.1. A B C D
7.2. A B C D
7.3. A B C D
7.4. A B C D
7.5. A B C D
7.6. A B C D

8. Branch Prediction and Exceptions and Interrupts

8.1. A B C D
8.2. A B C D
8.3. A B C D
8.4. A B C D
8.5. A B C D
8.6. A B C D

9. Input/Output

9.1. A B C D
9.2. A B C D
9.3. A B C D
9.4. A B C D
9.5. A B C D
9.6. A B C D

10. Caches

10.1. A B C D
10.2. A B C D
10.3. A B C D
10.4. A B C D
10.5. A B C D
10.6. A B C D

11. Virtual Memory

11.1. A B C D
11.2. A B C D
11.3. A B C D
11.4. A B C D
11.5. A B C D
11.6. A B C D

12. Parallelism

12.1. A B C D
12.2. A B C D
12.3. A B C D
12.4. A B C D
12.5. A B C D
12.6. A B C D

Feel free to detach the remaining pages of the exam and turn in only your answers.

1. ISA 1

1. What would the results of this program be if the ISA did not promise sequential and atomic execution?

```
R1=1, R2=2, R3=3
add R3, R2, R1
sub R2, R2, R2
```

2013-Dec-ISA 1-1

   a. R1=1, R2=0, R3=3
   b. R1=1, R2=0, R3=2
   c. A or B
   d. No way to know

If the processor doesn't promise sequential and atomic execution you have no way of knowing when the instructions will be executed. Not only could the order change (a or b) but the addition could be done half way before the next one is started, so you might not even finish the sub before starting the add!

2. Why do we need both a register read and an ALU operation to write to memory?

2013-Dec-ISA 1-2

   a. We don't
   b. The ALU always does something regardless of whether we use the results
   c. The register provides the write data and the ALU calculates the address
   d. The register provides the address and the ALU proves the write data

The sw instruction uses the ALU to calculate the address (adding the immediate offset to one of the registers) and uses the other register for the data to write.

3. What is the value of R2 after this MIPS code is executed? (ignore branch delay slots)

```
R2 = 4, R12=8
beq     R12, 17, skip
addi    R2, R0, 18
j       done
skip:
        addi    R2, R0, 20
done:
```

2013-Dec-ISA 1-3

   a. 18
   b. 20
   c. 22
   d. Program won't work

There is no beq R, constant, constant instruction! (Remember we don't have space for two immediates in an instruction.) Therefore the program is invalid and won't work.

4. This code does an unaligned load. What is the address of the word loaded into R1?

```
R5 = 12
lw R2, 0(R5)
lw R3, 4(R5)
sll R2, R2, 24
srl R3, R3, 8
or R1, R2, R3
```

2013-Dec-ISA 1-4

   a. 13
   b. 14
   c. 15
   d. 16

The code loads the words at addresses 12 and 16 into R2 and R3. It then shifts R2 left 24, which means R2 is left with 15|0|0|0. R3 is shifted to the right by 8, so it has 0|16|17|18. The two are then ORed to produce 15|16|17|18.

5. How many registers can you use to store data in a MIPS program?

2013-Dec-ISA 1-5

a. 32
b. 31
c. 24
d. Depends on which registers the function calls use

We can use any register except R0 to store data. If we use them in the wrong way then we may have compatibility issues, though.

6. How many values could you read from the register file in the time it takes to read from main memory on a MIPS processor? (It takes about 200 cycles to read from main memory on an these days.)
`2013-Dec-ISA 1-6`
    a. 1
    b. 200
    c. 400
    d. 600

On each cycle you can read two values from the register file, so in 200 cycles you could read the register file 400 times.

7. What address does the following code load?
```
addi R3, R0, 12
addi R2, R3, 12
addi R1, R0, 4
lw R1, 12(R2)
```
`2017-Jan-ISA 1-1`
    a. 4
    b. 12
    c. 24
    d. 36

The first line puts the constant 12 into R3
The second line adds 12 to the value in R3 and stores the result in R2 (So R2 now contains 24)
The third line puts the constant 4 into R1
The fourth line offsets the content of R2 by 12, and thus loads the address 24 + 12=36.

8. How many conditional branch instructions are needed to implement a for-loop that contains a nested if-then-else statement?
`2017-Jan-ISA 1-2`
    a. 1
    b. 2
    c. 3
    d. 4

A for-loop needs one conditional branch/check at the top of the loop after initialization, and one unconditional branch back to the beginning of the loop. An if-then-else statement requires one conditional branch/check for the if and an unconditional jump over the else if the initial conditional check fails. The result is 2 conditional branch instructions.

9. How many addressable registers can be used by the add instruction?
`2017-Jan-ISA 1-3`
    a. 31
    b. 32
    c. 33
    d. More than 33

There are 32 addressable registers. Even though R0 is not a fully usable register, it is addressable. There are other registers (PC, floating point, multiplication, etc.) but they are not general purpose.

10. Why is it less efficient to compute directly on data in memory than data in registers?
`2017-Jan-ISA 1-4`
    a. Registers are faster than main memory
    b. Registers require fewer address bits than main memory
    c. A and B

d. Computing on data in memory is as efficient as data in registers

The register file is much smaller than main memory so it is much faster to access (1 cycle vs 100 cycles) and we only need a few bits (5 bits for MIPS vs. 32 bits for memory) to address it.

11. Why do we want our processor to appear to be atomic and sequential?
`2017-Jan-ISA 1-5`
    a. Smaller
    b. Easier to build
    c. Faster
    d. Easier to understand

If the processor does not appear to operate in an atomic and sequential manner then we won't be able to understand how code will execute on it and we won't be able to program it.

12. How does the register file change after executing "addi R0, R1, 1"?
`2017-Jan-ISA 1-6`
    a. R0 has R1+1
    b. R0 has R0+R1+1
    c. R1 has R0+1
    d. None of the above

That instruction writes to R0, which does nothing.

13. How many bytes of arbitrary data can a program store in the MIPS register file?
`2014-Apr-ISA 1-1`
    a. 232
    b. 124
    c. 32
    d. 31

There are 31 register file entries that you can write to (R0 is always 0) and each register is 4 bytes (32 bits) so you can store 31*4=124 bytes in the register file.

14. Which of the following addresses are word-aligned?
`2014-Apr-ISA 1-2`
    a. 1110 1011 0000 1110
    b. 0100 0011 0000 1010
    c. A and B
    d. Neither

A word-aligned address has to be a multiple of 4, which means the last two bits need to be 00.

15. The following assembly code should implement if (a==b-c) {a=a+2;}. What should the missing instruction be? (a is in R5, b is in R6, and c is in R7)

```
sub     R2, R6, R7
_________________
addi    R5, R5, 2
skip:
```

`2014-Apr-ISA 1-3`
    a. beq R5, R2, skip
    b. beq R6, R2, skip
    c. j skip
    d. bne R5, R2, skip

R2 will have b-c so if R5==R2, then we know a==b-c and we want to do the addi. Therefore, if R5 !=R2, we skip it: bne R5, R2, skip.

16. What address does the following code load into R4?
```
lw R2, 16(R0)
sll R2, R2, 16
```

```
lw R3, 20(R0)
slr R3, R3, 16
or R4, R2, R3
```

2014-Apr-ISA 1-4

a. 17
b. 18
c. 19
d. 21

R2 first loads bytes [16,17,18,19] and is then shifted left by 16 bits to be [18,19,0,0]. R3 first loads bytes [20,21,22,23] and is shifted right by 16 bits to be [0,0,20,21]. When they are ored the result is [18,19,20,21].

17. How many sequencing instructions do you need to implement an if-than-else in MIPS assembly?

2014-Apr-ISA 1-5

a. 0
b. 1
c. 2
d. 3

You need a conditional branch to see if you should skip the if/else part and then an unconditional branch to skip the other part if you did the first one

18. How much space do branch labels (e.g., "skip:") take up in the code?

2014-Apr-ISA 1-6

a. None
b. 16 bits
c. 4 bytes
d. Depends on the length of the label

Branch labels are not part of the code. When the code is assembled the assembler converts the labels to the jump offset. (E.g., if the label "skip" is 4 instructions after the branch where it is called, then the assembler will insert a branch with a destination offset of +4.)

19. What address does the following code load?

```
addi R3, R0, 12
addi R2, R3, 12
addi R1, R0, 4
lw R1, 12(R2)
```

2015-Oct-ISA 1-1

a. 4
b. 12
c. 24
d. 36

The first line puts the constant 12 into R3
The second line adds 12 to the value in R3 and stores the result in R2 (So R2 now contains 24)
The third line puts the constant 4 into R1
The fourth line offsets the content of R2 by 12, and thus loads the address 24 + 12=36.

20. Why do we need both a register and an ALU to read data from memory?

2015-Oct-ISA 1-2

a. We don't
b. The ALU calculates the address and the register stores the data read
c. The ALU calculates the data and the register stores the address
d. The ALU always does something regardless of whether we use the results

Answer: B.

21. In MIPS, what would happen if you try to write into the R0 register?

2015-Oct-ISA 1-3

a. Your program might crash
b. Your operating system might crash
c. The content of R0 is changed

d. Nothing, this register always contains 0

Answer: D.

22. In MIPS, what special meaning does the "exit:" label has compared to other labels?
2015-Oct-ISA 1-4
    a. It has no special meaning
    b. It is used in branch instructions to jump to the end of a loop
    c. It is used in branch instructions to jump to the end of the program
    d. It marks the end of the current function

Labels only have the semantic you give them. You are free to choose whatever name you want..

23. If R4 contains an unsigned integer, what does the instruction "sll R4, R4, 2" do?
2015-Oct-ISA 1-5
    a. It multiplies the content of R4 by 4
    b. It shifts the bits in R4 2 positions to the left
    c. A and B
    d. Neither A nor B

sll stands for Shift Left Logical. It shifts the bit pattern in R4 N positions to the left. When working with unsigned integer, shifting the bit pattern N positions to the left is equivalent to multiplying by 2^N. Therefore, shifting R4 2 positions to the left is also equivalent to multiply it by 4.

24. Which of the following will NOT copy the contents of R2 to R1?
2015-Oct-ISA 1-6
    a. addi R1, R2, 0
    b. sll R1, R2, 0
    c. sw R1, R0(12)
       lw R2, R0(12)
    d. add R1, R2, R0

The sw/lw combination stores R1 into memory and then loads it into R2. This is equivalent to copying R1 to R2, not R2 to R1.

25. Which register is most likely to be modified?
2018-Oct-ISA 1-1
    a. R0
    b. A0
    c. RA
    d. PC

The program counter (PC) is modified every time an instruction is executed, so it is most likely to be modified.

26. What address does the following code load into R4?
```
addi R1, R0, 1
sll R1, R1, 3
addi R2, R0, 8
slr R2, R2, 2
or R4, R1, R2
```
2018-Oct-ISA 1-2
    a. 16
    b. 10
    c. 8
    d. None of the above

R1 gets 1. Then sll 3 multiplies it by 2^3=8, so we have R1=8. R2 gets 8. Then slr 2 divides it by 2^4, so we have R2=2. We then OR 8 and 2. Since they are both one hot, this is the same as adding them (1000 OR 0010 = 1010) which is 10.

27. How many addressable registers can be used by the add instruction?
2016-Aug-ISA 1-1

a. 31
b. 32
c. 33
d. More than 33

There are 32 addressable registers. Even though R0 is not a fully usable register, it is addressable. There are other registers (PC, floating point, multiplication, etc.) but they are not general purpose.

28. What are the source inputs for the bne instruction?
2016-Aug-ISA 1-2
    a. 1 register, one immediate
    b. 2 registers, one immediate
    c. 3 registers
    d. 1 register, two immediates

The bne instruction takes two register inputs, and one immediate input (for the offset).

29. How many conditional branch instructions are needed to implement a for-loop that contains a nested if-then-else statement?
2016-Aug-ISA 1-3
    a. 1
    b. 2
    c. 3
    d. 4

A for-loop needs one conditional branch/check at the top of the loop after initialization, and one unconditional branch back to the beginning of the loop. An if-then-else statement requires one conditional branch/check for the if and an unconditional jump over the else if the initial conditional check fails. The result is 2 conditional branch instructions.

30. If we extended MIPS to have each register hold 128-bits, how many bytes would be needed to store all of the data the register file?
2016-Aug-ISA 1-4
    a. 496
    b. 512
    c. 3,968
    d. 4,096

The register file will hold 31 * 128-bits = 3,968 bits of data (R0 is always 0). 3,968 bits is 496 bytes.

31. How many temporary registers (not counting the input and output registers) are needed to compute R5=(R5-R3)+(R3-R5)?
2016-Aug-ISA 1-5
    a. 0
    b. 1
    c. 2
    d. 3

Both R3 and R5 are needed for both computations. Therefore, we need to keep an extra copy of one of them in a temporary register to allow for the second computation. We can use an in-place add (the final operation) from R5 into R5 which will not result in an additional register being used.

32. How many registers do I-type instructions read?
2016-Aug-ISA 1-6
    a. 0
    b. 1
    c. 2
    d. Depends on the instruction.

While all I-type instructions have two register file fields, they read different numbers of instrutions. Bne, beq, and sw read two registers, while addi and lw read one but write one.

33. If we had 128 registers, 64-bit wide each, how many memory locations are needed to fill the register file?

2014-Dec-ISA 1-1

a. 127
b. 255
c. 1016
d. 1024

The register file can hold 254 words. (R0 is always 0). Each word is 4 bytes, and the memory is byte-addressed, so we need 254*4 or 1016 memory locations to fill the register file.

34. Which of the following statements is true?

2014-Dec-ISA 1-2

a. Instructions stored in memory take more space than data.
b. Instructions in memory cannot be written.
c. Instructions are interpreted differently than data.
d. Instructions and data are saved in different memories.

Both instructions and data are just bits saved in the same memory, them are just interpreted. You can read/write instructions with standard load/store instructions just like data, but if you corrupt them and then execute them your program will crash.

35. Which register/registers are more likely to be modified by an instruction?

2014-Dec-ISA 1-3

a. $zero
b. $v0-$v1
c. $ra
d. PC

The PC (program counter) is modified by every instruction to go on to the next instruction.

36. Which instruction should be placed at _____ to load a word starting at address 2 into R3?

```
lw R1, 0(R0)

__________
lw R2, 4(R0)
slr R2, R2, 16
or R3, R2, R1
```

2014-Dec-ISA 1-4

a. sll R1, R1, 8
b. sll R1, R1, 16
c. lbu R1, 16(R0)
d. slr R1, R1, 16

R1 contains the word starting at 0 = [0 1 2 3]. We need to shift it left 16 bits (2 bytes) to get [2 3 0 0], which we then OR with the word at address 4 = [4 5 6 7] shifted right 16 bits (2 bytes) = [0 0 4 5] to get [2 3 4 5].

37. Whenever the operating system switches between programs it has to store all of the current program's state (all the data stored inside the processor that the program is using) so that when it switches back it can restore it to where the program was with the data it needs. How many bytes are required to store the state we have used in our MIPS programs?

2014-Dec-ISA 1-5

a. 31
b. 32
c. 124
d. 128

We need to store 31 registers (no need to store R0 since it is always 0) plus the PC so we know which instruction to return to, for a total of 32 registers, or 128 bytes.

38. How many 32-bit data words do we need to execute a MIPS store instruction, and where do they come from?

2014-Dec-ISA 1-6

a. 1: from the register file
b. 1: from the immediate
c. 2: one from the register file and one from the immediate
d. 2: both from the register file

The store instruction needs an address and the data, both 32-bit words. This data comes from the two registers specified by the instruction, but we do get to add a 16-bit immediate to the address loaded from the register file.

39. Which of the following addresses is word-aligned?
2015-Apr-ISA 1-1
    a. 0110 1010 0110 1000 0110 1010 0110 1001
    b. 0000 0100 1000 1011 0110 1000 1100 1100
    c. 0000 0010 1000 1010 0110 1000 1110 1111
    d. 0000 0000 0000 1010 0110 1000 1010 1110

Word-aligned addresses must be multiples of 4, which means the last two bits are zero.

40. Why do we need both a register read and an ALU operation to write to memory?
2015-Apr-ISA 1-2
    a. We don't
    b. The register provides the write data and the ALU calculates the address
    c. The ALU always does something regardless of whether we use the results
    d. The register provides the address and the ALU proves the write data

The sw instruction uses the ALU to calculate the address (adding the immediate offset to one of the registers) and uses the other register for the data to write.

41. What address does the following code load?
```
addi R9, R0, 24?
addi R1, R9, 12
addi R9, R0, 12
lw R9, 12(R1)
```
2015-Apr-ISA 1-3
    a. 12
    b. 24
    c. 36
    d. 48

R9=24, R1=12+24=36, R9=12, load address = R1+12=48

42. Why do branch labels (e.g., "skip:") not take up any space in the program code?
2015-Apr-ISA 1-4
    a. We do not need a constant to jump to a label
    b. The label is included in instruction after the label
    c. We just use the label to know what constant to put in the branch/jump
    d. They do take up space!

Branch labels are not part of the code. When the code is assembled, the assembler uses the label to know what constant to put into the jump offset. (E.g., if the label "skip" is 4 instructions after the branch where it is called, then the assembler will use the constant +4 as the offset to get to the instruction.)

43. What value should be put in X to load the word starting at address 25 into R8?
```
lw R2, 24(R0)
sll R2, R2, 8
lw R3, 28(R0)
slr R3, R3, X
or R8, R2, R3
```
2015-Apr-ISA 1-5
    a. 8
    b. 16
    c. 24
    d. 32

R2 first loads bytes [24,25,26,27] and is then needs to be shifted left by 8 bits to be [25, 26, 27, 0]. R3 then loads [28,29,30,31] and needs to be shifted right by 24 bits to be [0,0,0,28]. The OR then creates [25,26,27,28].

44. Why would we ever execute an add with R0 as a source register?
2015-Apr-ISA 1-6

a. To move data between registers
b. To create a NOP
c. A and B
d. We wouldn't: it adds 0 so it is pointless

We use add dst, src, R0 to move data between registers and an add to R0 will have no effect so it can be used as a NOP.

45. If a word is 8 bytes, how many word addresses are there in a 32-bit byte-addressable computer?
`2013-Jan-ISA 1-1`
    a. 32
    b. 2^32
    c. (2^32)-8
    d. (2^32)/8

46. What is the minimum number of simultaneous reads and writes needed for a register file to work with the MIPS ISA?
`2013-Jan-ISA 1-2`
    a. 0 writes, 2 reads
    b. 1 write, 2 reads
    c. 1 write, 3 reads
    d. 2 writes, 3 reads

47. How does the register file change after executing "addi R0, R1, 1"?
`2013-Jan-ISA 1-3`
    a. R0 has R1+1
    b. R0 has R0+R1+1
    c. R1 has R0+1
    d. None of the above

48. The instruction "lui R1, 350" does what?
`2013-Jan-ISA 1-4`
    a. Loads the value at address 350 in to R1
    b. Loads the value 350 into R1
    c. Loads the value 350*2^16 into R1
    d. Stores the value 350 into the address in R1

49. How many temporary registers (not counting the input and output registers) are needed to compute R5=(R4-R3)+R2?
`2013-Jan-ISA 1-5`
    a. 0
    b. 1
    c. 2
    d. 3

50. What is required to load any unaligned memory address in MIPS?
`2013-Jan-ISA 1-6`
    a. A regular load word instruction
    b. Two load words, a shift, and an OR
    c. A load word, a load byte, a shift, and an OR
    d. You can't load data from unaligned memory addresses

51. Why is it less efficient to compute directly on data in memory than data in registers?
`2015-Aug-ISA 1-1`
    a. Main memory is slower than registers
    b. Main memory requires more address bits to access than registers
    c. A and B
    d. Computing on data in memory is as efficient as data in registers

The register file is much smaller than main memory so it is much faster to access (1 cycle vs 100 cycles) and we only need a few bits (5 bits for MIPS vs. 32 bits for memory) to address it.

52. Why do we want our processor to appear to be atomic and sequential?
`2015-Aug-ISA 1-2`
    a. Smaller

b. Easier to build
c. Faster
d. Easier to understand

If the processor does not appear to operate in an atomic and sequential manner then we won't be able to understand how code will execute on it and we won't be able to program it.

53. What address does the following code load?

```
addi R2, R0, 10?
addi R1, R2, 2
addi R2, R0, 12
lw R2, 24(R1)
```

2015-Aug-ISA 1-3

a. 12
b. 24
c. 36
d. 48

R2=10, R1=10+2=12, R2=12, load address = R1+24=36

54. What value should be put in X to load the word starting at address 26 into R8?

```
lw R2, 24(R0)
sll R2, R2, X
lw R3, 28(R0)
slr R3, R3, 16
or R8, R2, R3
```

2015-Aug-ISA 1-4

a. 8
b. 16
c. 24
d. 32

R2 first loads bytes [24,25,26,27] and is then needs to be shifted left by 16 bits to be [26,27,0,0]. R3 then loads [28,29,30,31] and needs to be shifted right by 16 bits to be [0,0,28,29]. The OR then creates [26,27,28,29].

55. How much space does a branch label take up in the machine code?

2015-Aug-ISA 1-5

a. 0 words
b. 1 word
c. 1 byte
d. 1 byte per character in the label name

Branch labels are not part of the code. When the code is assembled, the assembler uses the label to know what constant to put into the jump offset. (E.g., if the label "skip" is 4 instructions after the branch where it is called, then the assembler will use the constant +4 as the offset to get to the instruction.)

56. Which of the following MIPS registers can be written with a regular add instruction?

2015-Aug-ISA 1-6

a. R0
b. RA
c. PC
d. None can be written with a regular add

$ra is just register 31. We treat it specially with the calling convention, but you can use a regular add to change it. $r0 is always 0, so you can't change it. The PC can only be changed with branch and jump instructions.

57. Why do processors do all their computations on registers?

2014-Sep-ISA 1-1

a. Main memory is too slow
b. Easier to address fewer locations
c. A and B
d. None of the above

Accessing main memory on every instruction would take far too long and the number of address bits needed would make the instructions huge.

58. Can you write to the PC register?
`2014-Sep-ISA 1-2`
    a. Yes, same as any other register
    b. Yes, but you need to use special registers
    c. Yes, but not in a general purpose manner
    d. No

The PC is written on a jump or branch instruction. This means you can write it with JR instructions, but it's not easy to use it as a general purpose register.

59. What is this code equivalent to?
```
ld R1, (R2)
sll R1, R1, 24
slr R1, R1, 24
```
`2014-Sep-ISA 1-3`
    a. ld R1, 24(R2)
    b. ld R1, 3(R2)
    c. lbu R1, 3(R2)
    d. Need to know the value in R2 to determine

The first ld loads the value at R2 into R1. Now R1 has [A B C D]. The shift then shifts it to the left 24 bits, or 3 bytes. This leaves just the highest byte in R1: [D 0 0 0]. The shift to the right then puts that byte in the lowest byte: [0 0 0 D], which is equivalent to loading the byte at address R2+3.

60. How many memory locations are needed to fill the register file?
`2014-Sep-ISA 1-4`
    a. 31
    b. 32
    c. 124
    d. 128

The register file can hold 31 words. (R0 is always 0). Each word is 4 bytes, and the memory is byte-addressed, so we need 31*4 or 124 memory locations to fill the register file.

61. What is the difference between instructions and data stored in memory?
`2014-Sep-ISA 1-5`
    a. They are stored in different memories
    b. They can only be accessed with different instructions
    c. Instructions cannot be written while data can
    d. There is no difference

Both instructions and data are just bits in memory. What makes them different is how they are interpreted. You can read/write instructions with standard load/store instructions just like data, but if you corrupt them and then execute them your program will crash.

62. Why is the PC incremented by 4?
`2014-Sep-ISA 1-6`
    a. Each instruction is 4 bytes long and the memory is word-addressed
    b. Each instruction is 4 bytes long and the memory is byte-addressed
    c. The increment depends on the address for conditional jumps
    d. The last two bits of the instruction are always 00

Since each instruction is 4 bytes long we need to move ahead by 4 bytes (one instruction) for each next instruction. The memory is byte-addressed, which means that we need to send it a byte address, so each instruction is 4 bytes ahead of the next. The conditional jumps do add 4 and the last two bits of the instruction are not always 00. (The last two bits of the instruction's address are always 00.)

63. How many of the least significant bits of a word address must be zero in a byte-addressable machine that only loads 32-bit aligned words?
`2016-Oct-ISA 1-1`

a. 2
b. 3
c. 4
d. None, the machine is byte-addressable

2 bits will cover 4 bytes, or one word.

64. What does the following code put in R3?

```
lw R1, 12(R0)
lw R2, 16(R0)
sll R1, R1, 24
slr R2, R2, 8
or R3, R1, R2
```

2016-Oct-ISA 1-2

a. Zero
b. The ORed sum of the data at address 12 and 16
c. The byte at address 16
d. The word starting at address 15

The loads bring in data to R1=[12 13 14 15] and R2=[16 17 18 19]. R1 is then shifted 24 to the left [15 0 0 0] and R2 is shifted 8 to the right [0 16 17 18]. They are then combined to have bytes [15 16 17 18].

65. What address does the following code load?

```
addi R2, R0, 24
addi R3, R2, 4
lw R2, 4(R3)
```

2016-Oct-ISA 1-3

a. 24
b. 28
c. 32
d. 112

R2=24, R3=24+4=28, and the address is R3+4=32.

66. From where do immediate instructions get their immediate values?

2016-Oct-ISA 1-4

a. The register file
b. The instruction
c. The ALU
d. Depends on the instruction type

Immediate values are encoded in the instruction itself

67. How many control flow instructions are needed to implement an if-then-else statement?

2016-Oct-ISA 1-5

a. 1
b. 2
c. 3
d. 2-3, depending on the conditional

Two. You need a BNE or similar instruction, and a J to skip over the else portion.

68. How many values could you write to the register file in the time it takes to read from main memory on a MIPS processor? (It takes about 200 cycles to read from main memory)?

2016-Oct-ISA 1-6

a. 1
b. 200
c. 400
d. 600

In 200 cycles you can write 200 values to the register file. (1 per cycle.)

69. What does the lw instruction need to work?
2016-Jan-ISA 1-1
    a. An immediate for the address
    b. An immediate for the address and a register for the data
    c. A register for the address and an immediate for the data
    d. A register and an immediate for the address and a register for the data

lw calculates the address as imm+reg and then uses the other reg as the destination of the returned data from memory.

70. What is the minimum number of simultaneous reads and writes for a register file to work with the MIPS ISA?
2016-Jan-ISA 1-2
    a. 1 write, 1 read
    b. 2 writes, 2 reads
    c. 1 write, 2 reads
    d. 2 writes, 1 read

R-type instructions read two register file entries and write to a third.

71. How many executed instructions would it take to increment each program value in the MIPS register file?
2016-Jan-ISA 1-3
    a. 31
    b. 32
    c. 62
    d. 93

The register file has 32 entries, but you can only change 31 of them. For each you need an addi to increment it by 1 and store it back, so you need 31.

72. How many control instructions are required in addition to beq to implement an if-then-else in MIPS assembly?
2016-Jan-ISA 1-4
    a. 0
    b. 1
    c. 2
    d. Depends on whether you count the branch delay slot

The beq can jump to either the else or the if, but you still need a j to skip the else/if when you are in the other branch.

73. What address does the following code load?
```
addi R1, R0, 4
addi R0, R0, 4
add R2, R0, R1
lw R3, 4(R2)
```
2016-Jan-ISA 1-5
    a. 4
    b. 8
    c. 12
    d. 16

The first two lines store 4 into R1 and try to store 4 into R0, which doesn't matter since R0 is always zero. R2 is then calculated as 0+4, or 4, and the load word loads from an offset of 4 off from R2, or 8..

74. What values do you need to put into X and Y to load the word at address 27 into R2?
```
lw R1, 24(R0)
lw R2, 28(R0)
sll R1, R1, X
slr R2, R2, Y
or R2, R1, R2
```
2016-Jan-ISA 1-6
    a. X=0, Y=32
    b. X=16, Y=16
    c. X=24, Y=8

d. X=8, Y=24

R1 initially loads addresses [24 25 26 27] and needs to be shifted 24 bits to the left to have [27 0 0 0]. R2 initially loads [28 29 30 31] and needs to be shifted 8 bits to the right to have [0 28 29 30]. They can then be ORed to produce [27 28 29 30]..

75. Which address would be an aligned address on a machine that only loads 64-bit words with a byte-addressable memory?
`2013-Aug-ISA 1-1`
    a. 2
    b. 4
    c. 8
    d. 12

64-bit words have 8 bytes, so for a byte-addressable memory you need to have the last 3 bits 0 to be aligned to an 8-byte boundary. 8=1000, but 12=1100.

76. How many control flow instructions are needed to implement a simple for-loop?
`2013-Aug-ISA 1-2`
    a. 1
    b. 2
    c. 3
    d. Depends on whether you end the loop at 0

1: one conditional branch can be used to jump back to the start of the loop until it is finished, and then you just continue past the branch when done. You need two for an if-then-else statement.

77. Why does the PC need information from the ALU?
`2013-Aug-ISA 1-3`
    a. It uses the ALU to do PC+4
    b. It uses the ALU to load the instruction
    c. It uses the ALU to decide the next instruction
    d. All of the above

The ALU is used to calculate whether conditional branches should be taken. The PC+4 and the instruction load are not done by the ALU.

78. How many fully usable general-purpose registers does MIPS have?
`2013-Aug-ISA 1-4`
    a. 31
    b. 32
    c. 33
    d. More than 33

31. There are 32 addressable registers, but R0 is not fully usable because you can't write data to it. There are other registers (PC, floating point, multiplication, etc.) but they are not general purpose.

79. How many temporary registers are required to compute A-(B+C) if you need A, B, and C later?
`2013-Aug-ISA 1-5`
    a. 0
    b. 1
    c. 2
    d. 3

1. add t, B, C; sub t, A, t

80. From where do immediate instructions get their immediate values?
`2013-Aug-ISA 1-6`
    a. The register file
    b. The instruction
    c. The ALU
    d. Depends on the instruction type

The instruction. Immediate values are encoded in the instruction itself.

81. Which of the following addresses is word-algined?
2013-Apr-ISA 1-1
    a. 0110 1010 0110 1000 0110 1010 0110 1001
    b. 0000 0000 1000 1010 0110 1000 1010 1111
    c. 0000 0000 0000 1010 0110 1000 1010 1110
    d. 0000 0000 1000 1010 0110 1010 1010 1100

Word-aligned addresses must be multiples of 4, which means the last two bits are zero.

82. What is the value in R1 after the following program? (starts with R0=0, R1=1, R2=2)
```
add R1, R1, R2
add R0, R0, R1
add R1, R0, R1
```
2013-Apr-ISA 1-2
    a. 0
    b. 2
    c. 3
    d. 6

Solution: R1<--1+2 = 3, then R0<--0+3, but R0 is always zero. So finally R1<--0+3=3

83. Why do we want our processor to appear to be atomic and sequential?
2013-Apr-ISA 1-3
    a. Faster
    b. Easier to build
    c. Easier to understand
    d. Smaller

If the processor does not appear to be atomic and sequential we will not be able to figure out how it will execute instructions and we won't be able to understand it.

84. What does the following code put in R3?
```
lw R1, 12(R0)
lw R2, 16(R0)
sll R1, R1, 16
slr R2, R2, 16
or R3, R1, R2
```
2013-Apr-ISA 1-4
    a. Zero
    b. The ORed sum of the data at addresses 12 and 16
    c. The ORed sum of the data at addresses 12 and 16 multiplied by 216
    d. The word starting at address 14

This code loads the data at word-aligned addresses 12 and 16 into R1 and R2, then shifts R1 and R2 such that R1 has the data at bytes 14,15,00000... and R2 has 0000...,16,17. The two are then ORed to produce the un-aligned word from address 14, 15, 16, 17.

85. What address does the following code load?
```
addi R2, R0, 12
addi R3, R2, 24
lw R2, 12(R3)
```
2013-Apr-ISA 1-5
    a. 12
    b. 24
    c. 36
    d. 48

R2=12, R3=12+24=36, load address = R3+12=48

86. What instructions go at XXX and YYY to accomplish: if (R3==R4) then R5=2 else R5=1?
```
XXX R3, R4, labelA
```

```
addi R5, R0, 1
YYY R0, R0, labelB
labelA:
addi R5, R0, 2
labelB:
```

2013-Apr-ISA 1-6

a. XXX=bne YYY=j
b. XXX=beq YYY=j
c. XXX=bne YYY=beq
d. XXX=beq YYY=beq

We want to jump to labelA (R5=2) if R3==R4, so XXX=beq. We want to skip over labelA all the time if we do the first addi, so YYY=beq (since R0 always equals R0).

2. ISA 2

1. How does the processor know the instruction type?

   2013-Dec-ISA 2-1

   a. The binary code contains an instruction type bit for each instruction
   b. If there is an immediate in the instruction
   c. Looks at the opcode field
   d. If the instruction has 0, 2, or 3 register fields

   The processor has to look at the opcode field and decode it. There is no way to know if the bits in the instruction are immediates or register fields without decoding the opcode to know the type.

2. Why is the immediate field shifted left for branch instructions?

   2013-Dec-ISA 2-2

   a. Instruction addresses are always multiples of 4
   b. Instructions are word-aligned
   c. By shifting it left by two we can increase the jump range by 4x
   d. All of the above

   A and b are really the same thing since a word is 4 bytes.

3. If we changed our 32-bit processor to have 256 registers, how many instructions would we need to load a 32-bit constant if we keep similar I, R, and J format instructions.

   2013-Dec-ISA 2-3

   a. 4
   b. 3
   c. 2
   d. 1

   With 256 registers we need 8 bits per register. This means the I-type instructions will need 16 bits for the register values, which is 6 more than the 10 bits they have with 32 registers. This would reduce the size of their immediate field to 10 bits, which would require 4 immediate constants to fill up a 32 bit constant.

4. This method might be familiar to you if you recall the first lab assignment: the method1 function receives a pointer to an array of characters (String) and iterates over them until it reaches the end (last byte is 0). For each element of the array, method1 calls an additional function (method2) to execute some computation on that element. The result of this computation is then captured by method1 and stored in the array, replacing the previous data in that position.

```
method1:
        addi    $t0, $a0, 0
loop:
        lb      $s0, 0($t0)
        beq     $s0, $zero, end
        addi    $a0, $t0, 0
        jal     method2
        sb      $v0, 0($t0)
        addi    $t0, $t0, 1
        j       loop
end:
        jr      $ra
```

What do we need to save and restore for this code to follow the MIPS calling convention? (There are two places we need to save and restore: at the start of method1 and before returning from method1, and before calling method2 and after returning from it.)

2013-Dec-ISA 2-4

a. method1: $t0, method2: $ra
b. method1: $t0, $s0, method2: $ra
c. method1: $s0, method2: $t0
d. method1: $s0, method2: $t0, $ra

Both $t0 and $ra are caller saved, so they must be saved before calling method2 if we want to preserve them (and we do since we use them later). $s0 is callee saved, which means that if we overwrite it, we must restore the previous value. Method1 does overwrite $s0, so we need to save and restore it before returning.

5. Here is a small snippet of code that saves $t0 and $t1 to the stack. What instruction should be put in the place of X?

```
X
sw      $t0, 0($sp)
sw      $t1, 4($sp)
```

2013-Dec-ISA 2-5

a. addi $sp, $sp, 4
b. addi $sp, $sp, -4
c. addi $sp, $sp, 8
d. addi $sp, $sp, -8

We are storing two words to the stack (8 bytes total). This means we need to move the stack pointer by 8 bytes. Since the stack grows down, we need to subtract 8.

6. What registers does this method need to save and restore when it starts and returns to follow the MIPS calling convention?

```
method3:
        addi    $t0, $zero, 0
        beq     $a0, $zero, end
        sll     $t0, $a0, 2
        add     $t0, $t0, $a1
end:
        addi    $vo, $t0, 0
        jr      $ra
```

2013-Dec-ISA 2-6

a. None
b. $t0
c. $t0, $a0, $a1
d. $t0, $a0, $a1, $ra

This method does not call any other methods so we do not need to save any of the registers it uses.

7. main() uses t0, t1, t2, s0, B() uses t4, s3, s4, C() uses t1, t2, t3, t4, s0, s5. How many words on the stack are needed when main() calls B() and B() calls C()?

2017-Jan-ISA 2-1

a. 8
b. 9
c. 11
d. 17

main() is only a caller, so it saves: t0, t1, t2, B is a caller and a callee, so it needs to save: t4, s3, s4, ra; and C is only a calle, so it saves: s0, s5 = 9 total

8. What problem does the stack solve?

2017-Jan-ISA 2-2

a. Storing data before loading it into a register and after removing it from a register
b. There are not enough registers for each procedure to have its own
c. Procedures do not know which registers their caller and callees use

d. How to access more than 32 words of data

The stack allows us to store more values than we can fit in the register file. If we had an infinite register file then each procedure could use its own registers and we wouldn't need the stack.

9. Which of the following is NOT a valid statement?
`2017-Jan-ISA 2-3`
   a. beq R1, R5, exit
   b. add RA,R0,T1
   c. sub R0,R0,R0
   d. addi T1,T2,82000

The immediate is a signed 16 bit integer, so this value can not be represented (max is 215 -1 or 32767).

10. How does the processor know the instruction type?
`2017-Jan-ISA 2-4`
   a. If the instruction has 0, 2, or 3 register fields
   b. If there is an immediate in the instruction
   c. The binary code contains an instruction type bit for each instruction
   d. None of the above

The processor has to look at the opcode field and decode it. There is no way to know if the bits in the instruction are immediates or register fields without decoding the opcode to know the type.

11. A designer adds a "load big constant to register" instruction to MIPS: "lbc immediate" that always stores the result into R1. How much does this speed up loading a 32-bit immediate?
`2017-Jan-ISA 2-5`
   a. No change
   b. 33% faster
   c. 1.5x faster
   d. 2x faster

To encode this instruction we need just the opcode since the register is defined as part of the instruction. Therefore we can get 32-6 = 26 bits of immediate from each instruction, which doesn't save us any instructions over loading 16 bits per instruction. (We still need two, a shift, and an OR.)

12. This is the last code in a procedure. What instructions should be placed in _____?
lw $t0, 0($sp)
lw $t1, 4($sp)
_____
_____
`2017-Jan-ISA 2-6`
   a. addi $sp, $sp, 8; jr $ra
   b. addi $sp, $sp, -8; jal proceedure
   c. addi $sp, $sp, 2; jr $ra
   d. addi $sp, $sp, -2; jal proceedure

We are loading two words from the stack back into $t0 and $t1, so we need to move the stack pointer back UP by 2 words, or 8 bytes, and then do a return jump.

13. What size adder do we need to include the immediate into the next PC for a jump instruction?
`2014-Apr-ISA 2-1`
   a. None
   b. 4 bit
   c. 26 bit
   d. 32 bit

None. The jump instruction just replaces 26 bits of the PC with the 26 bit immediate from the instruction.

14. How would you conditionally jump 1 million instructions forward in a program?
`2014-Apr-ISA 2-2`
   a. Use a standard bne/beq instruction

b. Use a standard j instruction
c. Use a bne/beq instruction to control whether you execute a j instruction
d. You can't because conditional branches only have one constant

First use a bne/beq to decide if you should jump, and then have that instruction jump to the j instruction to do the longer jump. Remember that beq/bne have a 16 bit constant, which means they can only jump +/- about 32k instructions.

15. Which registers need to be saved in my_procedure?

```
my_procedure:
addi $t1, $a0, 4
add $s0, $t1, $a1
sub $a2, $s0, $a1
jal fancy_math
sub $v0, $t2, $v1
jr $ra
```

2014-Apr-ISA 2-3

a. $s0, $ra
b. $s0, $t1, $ra
c. $s0, $t2, $ra
d. $s0, $t1, $t2, $ra

$s0 is a callee save register so we need to save it in case our caller needs it. $t2 is a caller save register and fancy_math may overwrite it, so we need to save that. $ra will be overwritten when we call fancy_math, so we need to save that. We do not need to save $t1 because even if fancy_math overwites it we do not reuse it.

16. We used the lui (load upper immediate) and the ori (or immediate) instructions to load a full 32-bit constant. What can you say about the sign extension of the immediate field?

2014-Apr-ISA 2-4

a. All immediates are sign-extended
b. At least the ori instruction must not sign-extend its immediate
c. Can't conclude anything since sign-extending as part of an OR won't matter
d. This code example is too simple to tell us anything

For the ori instruction we must not sign extend the immediate field. If the MSB was 1, we would end up ORing in 1s into the top 16 bits which would replace whatever was already there with 1s

17. Imagine a new MIPS instruction: bnei REG, CONST, ADDRESS which branches to ADDRESS if REG!=CONST in the same way that bne does. How many bits would this instruction require to encode? (You do not have to keep to the same instruction format as MIPS, but the decoding should be similar.)

2014-Apr-ISA 2-5

a. 16
b. 32
c. 48
d. Depends on the size of the constants

This depends on the size of the constants. If we want two 16 bit constants (one for both the 7 constant and the skip constant) then we need 16 bits more or 48 bits. But we could reduce the size, and only allow 8 bits for both and fit it in the 32 bit size.
Answer

18. Imagine we add a new MIPS instruction to the MIPS architecture discussed in class: bnei REG, CONST, ADDRESS which branches to ADDRESS if REG!=CONST in the same way that bne does. If we allow an 8 bit CONST, how far can the branch jump?

2014-Apr-ISA 2-6

a. +31 / -32 instructions
b. +64 / -64 instructions
c. +127 / -128 instructions
d. +256 / - 256 instructions

With 8 bits for the CONST we will have 8 bits left over for the ADDRESS offset. The address is encoded as a two's complement instruction offset just like bne, so this will be between +127 and -128 instructions.

19. Which statement is NOT true?
2015-Oct-ISA 2-1
    a. J can only jump within its 256 MB (223) regions
    b. BEQ can sometimes jump (branch) further ahead than J can
    c. Either J or BEQ can always jump further than JR
    d. BEQ and BNE have the same restrictions for the jump target

JR can specify a full 32 bit address, and have thus no restrictions on where to jump

20. main() uses t0, t1, t2, s0, B() uses t4, s3, s4, C() uses t1, t2, t3, t4, s0, s5. How many words on the stack are needed when main() calls B() and B() calls C()?
2015-Oct-ISA 2-2
    a. 8
    b. 9
    c. 11
    d. 17

main() is only a caller, so it saves: t0, t1, t2, B is a caller and a callee, so it needs to save: t4, s3, s4, ra; and C is only a calle, so it saves: s0, s5 = 9 total

21. Which of the following is NOT a valid statement?
2015-Oct-ISA 2-3
    a. beq R1, R5, exit
    b. add RA,R0,T1
    c. sub R0,R0,R0
    d. addi T1,T2,63000

The immediate is a signed 16 bit integer, so this value can not be represented (max is 215 -1 or 32767).

22. What is written to $RA when the JR instruction is executed?
2015-Oct-ISA 2-4
    a. Nothing
    b. PC
    c. PC + 4
    d. The previous value of the $RA register

Nothing. However, the value is read.

23. What is a disadvantage of a load-store register machine compared to a memory-register machines?
2015-Oct-ISA 2-5
    a. More complex instruction decoder
    b. Cannot be pipelined
    c. Less dense code
    d. Denser code

You generally need more instructions to achieve the same thing with a load-store isa.

24. When calling another procedure, a certain convention must be followed. Why?
2015-Oct-ISA 2-6
    a. To minimize the stack size
    b. To avoid corrupting data from other programs.
    c. Procedures do not know which registers their caller and callees use
    d. There may be an arithmetic overflow, which must be checked beforehand

If you knew exactly which registers would and wouldn't be used you wouldn't need a convention, but then you couldn't have libraries either.

25. A dual-issue pipeline has one ALU that can take inputs from two registers or one register and the sign-extender, and sends its result back to the register file. The second ALU can take inputs from one register and the sign-extender, but only sends its result to the address input of the memory. A designer realizes that this is not ideal and changes it so either ALU can send their results back to the register file. Which pair of instructions can now be executed at the same time that could not

be before? (Note, the list below does not imply any ordering. E.g., A and B does not mean that A has to execute on ALU 1 and B on ALU 2.)

2018-Oct-ISA 2-1

a. add and lw
b. addi and lw
c. add and addi
d. lw and lw

[ALU1=add, ALU2=lw] could be done before. [ALU1=addi, ALU2=lw] could be done before. [ALU1=lw, ALU2=lw] cannot be done on either since both would need to write to the memory at the same time. With the new design, [ALU1=add, ALU2=addi] is now possible since addi writes back to the register file.

26. Why do we decrement the stack pointer when we store data to the stack?

2016-Aug-ISA 2-1

a. To prevent the stack from running into program memory
b. Because memory addresses grow down
c. Because it is a requirement by the ISA
d. All of the above.

This is just an arbitrary choice for the ISA: the stack grows down (and the heap grows up). As long as all methods use the same standard everything works together.

27. Which instruction uses the immediate field of the data directly without the need to change the value (ex. shift the value, add value(s), etc.)?

2016-Aug-ISA 2-2

a. bne
b. j
c. ori
d. lui

All instructions modify the meaning of the immediate field. The bne instruction first computes PC+4+(imm << 2), j modifies the immediate by (imm << 2), and lui shifts the immediate by 16 bits (imm << 16). The ori instruction does not modify the value of the immediate field.

28. What are the advantages of an encoding where some instructions can use fewer bits than others?

2016-Aug-ISA 2-3

a. Smaller instructions
b. Larger potential for size of immediate fields
c. Ease to extend the ISA
d. All of the above

Inconsistent instruction encodings (like x86 or IBM's zSeries) allows for smaller, more compact code and potential extensions of the ISA by using longer instructions. Through ISA extensions, it is possible to have larger immediate fields as well.

29. A designer adds a "load big constant to register" instruction to MIPS: "lbc immediate" that always stores the result into R1. How much does this speed up loading a 32-bit immediate?

2016-Aug-ISA 2-4

a. No change
b. 33% faster
c. 66% faster
d. 2x faster

To encode this instruction we need just the opcode since the register is defined as part of the instruction. Therefore we can get 32-6 = 26 bits of immediate from each instruction, which doesn't save us any instructions over loading 16 bits per instruction. (We still need two, a shift, and an OR.)

30. What does the standard MIPS calling convention tell a programmer?

2016-Aug-ISA 2-5

a. How to access more than 32 bits of data
b. How to store data before loading it into a register and after removing it from a register
c. That there are not enough registers for each procedure to have its own
d. That procedures do not know which registers their caller and callees might use

The calling convention tells us how to handle registers to make sure we do not overwrite registers other parts of the program are using (caller) or have other parts overwrite ours (callee), without knowing specifically which ones they are using..

31. How does the processor know the instruction type?
    2016-Aug-ISA 2-6
    a. If the instruction has 0, 2, or 3 register fields
    b. If there is an immediate in the instruction
    c. The binary code contains an instruction type bit for each instruction
    d. None of the above

    The processor has to look at the opcode field and decode it. There is no way to know if the bits in the instruction are immediates or register fields without decoding the opcode to know the type.

32. To make procedure calls faster, a designer decides to change the JAL instruction to increment the stack pointer by 2 words automatically and JR to decrement the stack pointer by 2 words automatically. The designer adds special hardware to the processor to do this in the same time as the regular JAL and JR instructions. What effect does this have?
    2014-Dec-ISA 2-1
    a. Procedures that need to save 1 or 2 registers are faster
    b. Procedures that need to save more than 2 registers are not faster
    c. Procedures that save fewer than 2 registers waste space on the stack
    d. All of the above

    For procedures that save 1 or 2 words, this optimization avoids having to change the stack pointer before putting data on the stack and after taking it off, which will speed up the procedures. If you need more than 2, however, you still have to do this. For procedures that don't use 2 positions on the stack, this approach will always take up the memory space, which will waste space.

33. A designer decides to add a fused multiply-add (FMA) instruction to our MIPS processor. The instruction does the following operation on registers: A=A*B+C. What type of instruction format can we use to encode this new instruction?
    2014-Dec-ISA 2-2
    a. R
    b. I
    c. J
    d. Need a new format

    We need three registers so we can use the R format. Note that A is both a source and the destination, so we would need a third read port on our register file.

34. How do variable-length instructions (used in Intel's x86 processors) compare to the fixed-length instructions MIPS uses?
    2014-Dec-ISA 2-3
    a. Smaller, easier to decode
    b. Smaller, harder to decode
    c. Larger, harder to decode
    d. Larger, easier to decode

    Variable length instructions can be a lot smaller (common instructions use fewer bytes) but they are much harder to decode because the processor has to figure out where each instruction ends.

35. How long will this program take to get to finish? (Assume each instruction takes 1 cycle; e.g., no delay slots.)

```
start:
  jal middle
  finish:
middle:
  jal last
  jr $ra
last:
  jr $ra
```

    2014-Dec-ISA 2-4
    a. 3 cycles
    b. 4 cycles
    c. 8 cycles

d. Forever

Since middle does not store $ra on the stack before it calls last, when we return from last we will have the address of the jr in middle stored in $ra, so jr $ra in middle will jump right back to jr $ra in middle, and repeat forever.

36. A designer decides that having a branch-if-equal-to-register-plus-constant instruction is a great idea and implements it by modifying the existing branch instruction to use half of the immediate field for the constant, encoded in the same way as the address. How many cycles would this new instruction save in the code below? (Assume no branch delay slots.)

```
procedure:
addi $t0, $a0, 255      // Check inputs to the procedure
beq $t0, $a1, invalid   // Branch based on the constant calculation
  addi $v0, $0r, 1      // Valid input, return 1
  jr $ra
invalid:
  addi $v0, $r0, 0      // Invalid input, return 0
  jr $ra
```

2014-Dec-ISA 2-5

a. 0
b. 1
c. 2
d. Depends on the values of $a0 and $a1

This new instruction has 8 bits for the address of the branch and 8 bits for the constant, both encoded as two's compliment numbers. This means it can jump -128 to +127 instructions and compare to a constant from -128 to +127. Since this instruction needs a constant of 255 we can't use it here, so it saves nothing.

37. Which kind of branches or jump can you use to get from the instruction at address A to the instruction at address B?
A: 01100000 00000000 00000000 00001100
B: 01110000 00000000 00000000 00010000

2014-Dec-ISA 2-6

a. ONLY beq/bne
b. ONLY j
c. Can use EITHER beq/bne OR j
d. Need BOTH beq/bne AND j

j only changes the lower 26 bits (the last 2 are always 0). So to change the upper 4 we need to use a branch to add. For this jump we would have to first do j 11 11111111 11111111 11111111 which would take us to 01101111 11111111 11111111 11111111 and then use a branch to move 9 instructions ahead to the final address.

38. Why is the instruction "bne R9, 1, loop" not possible in the MIPS ISA encoding?

2015-Apr-ISA 2-1

a. Because R-type instructions only have enough immediate field bits for one constant
b. Because I-type instructions don't have enough immediate field bits for two constant
c. Because I-type instructions need a constant offset, not a string like "loop"
d. Because J-type instructions don't have any bits for registers

The I-type instructions use X bits for the opcode, and from the remaining Y bits, Z are for the register, and there is only XX left for the constants.

39. How would you load the value 9 into the PC?

2015-Apr-ISA 2-2

a. lui $pc, 0
b. addi $pc, $r0, 9
   jr $pc
c. lui $pc, 9
d. addi $ra, $r0, 9
   jr $ra

You can't write directly into the PC, but the jr instruction will load the value from register $ra into the PC.

40. What happens with the immediate for branch instructions, and why?

2015-Apr-ISA 2-3

a. It is shifted right by 8 bits because the lower byte is not necessary
b. It is shifted right because instruction addresses are always multiples of 4
c. It is shifted left because instructions are word-aligned
d. It is not modified

A and b are really the same thing since a word is 4 bytes.

41. How can the processor know if an instruction is I-type and not R-type?
2015-Apr-ISA 2-4
a. It looks at the opcode
b. It looks if there is an immediate field
c. It looks if the destination register is in the right place
d. All of the above

The opcode tells the processor the type of instruction. The rest of the instruction is interpreted according to the opcode. E.g., the immediate field is only an immediate if the opcode says it is an I-type, otherwise it is part of the register fields and the func field.

42. This code will never finish. Why?

```
main:
        addi    R1, R0, 4
        addi    R2, R0, 0
        jal     sum_from_next_address
        j       finish
sum_from_next_address:
        addi    R1, R1, 4
        jal     load_data
        add     R2, R2, R3
        jr
load_data:
        lw      R3, 0(R1)
        jr
```

2015-Apr-ISA 2-5
a. It doesn't save temporary and saved registers
b. It doesn't save the return address register
c. It doesn't follow the register convention for argument and return registers
d. All of the above

This code doesn't save temporary and save registers nor follow the register convention, but it doesn't need to do what it says. However, because it doesn't save the return address when calling jal load_data, it will never return back to main.

43. Why do we decrement the stack pointer when we store data to the stack?
2015-Apr-ISA 2-6
a. Because the ISA defines it that way
b. Because memory addresses go down
c. To avoid running out of space if we increment it
d. All of the above

This is just an arbitrary choice for the ISA: the stack grows down (and the heap grows up). As long as all methods use the same standard everything works together.

44. What does the "jal" instruction do?
2013-Jan-ISA 2-1
a. First saves the address of the next instruction in a user-specified register, and then jumps to the given address
b. First jumps to the given address, and then saves the address of the next instruction in the $ra register
c. First saves the address of the next instruction in the $ra register, and then jumps to the given address
d. First jumps to the given address, and then saves the address of the next instruction in a user-specified register

45. Why is the instruction "bne R2, 5, loop" not possible in the MIPS ISA?
2013-Jan-ISA 2-2
a. Because I-type instructions don't have enough immediate field bits for two constants

b. Because R-type instructions only have enough immediate field bits for one constant
c. Because J-type instructions don't have any bits for registers
d. Because I-type instructions need a constant offset, not a string like "loop"

46. A J-type instruction has 6-bits for the opcode and 26-bits of immediate field. It works by:
2013-Jan-ISA 2-3
a. Using the immediate as a signed offset to the current PC (imm <<2 + PC)
b. Using the immediate as a signed offset to the next PC (imm<<2 + PC+4)
c. Replacing the lower bits of the next PC directly (imm <<2 overwrites bits 2 to 28 of PC+4)
d. Directly using the immediate value for the new PC (next PC is the immediate value)

47. What is the value of $ra right after instructions 8, 24, and 34?

```
0:  addi R1, $zero, 5
4:  addi R2, $zero, 7
8:  jal  process
12: j    exit
16: process:
     add  R3, R1, R2
20:  sub  R4, R1, R2
24:  jal  add
28:  jr   $ra
32: add:
     addi R3, R1, 2
34:  jr   $ra
```

2013-Jan-ISA 2-4
a. 8, 24, 34
b. 8, 24, 24
c. 12, 28, 12
d. 12, 28, 28

48. For question 4 above, what should the value of $ra be right after instructions 8, 24, and 34 such that the program executes instruction 12 after finishing process and add?
2013-Jan-ISA 2-5
a. 8, 24, 34
b. 8, 24, 24
c. 12, 28, 12
d. 12, 28, 28

49. How does the code need to be modified to make it exit correctly?
2013-Jan-ISA 2-6
a. Store $ra on the stack before instruction 8 and restore it before instruction 12
b. Store $ra on the stack before instruction 24 and restore it before instruction 28
c. Store $ra on the stack before instructions 28 and 34, and restore it before instructions 32 and 36
d. Use the caller/callee register saving conventions for R1, R2, R3, and R4

50. What problem does the stack solve?
2015-Aug-ISA 2-1
a. Storing data before loading it into a register and after removing it from a register
b. There are not enough registers for each procedure to have its own
c. Procedures do not know which registers their caller and callees use
d. How to access more than 32 words of data

The stack allows us to store more values than we can fit in the register file. If we had an infinite register file then each procedure could use its own registers and we wouldn't need the stack.

51. What problem does the MIPS calling convention solve?
2015-Aug-ISA 2-2
a. Storing data before loading it into a register and after removing it from a register
b. There are not enough registers for each procedure to have its own
c. Procedures do not know which registers their caller and callees use
d. How to access more than 32 words of data

The calling convention tells us how to handle registers to make sure we do not overwrite registers other parts of the program are using (caller) or have other parts overwrite ours (callee), without knowing specifically which ones they are using.

52. The following code is part of a library and may be called by other programs. What registers does the compute function need to save on the stack for it to work correctly?

```
begin:
add $t0, $t1, 3
ld $a0, 16($t0)
jal compute
add $s0, $v0, $t0
... (program ends eventually)
compute:
add $s1, $a0, $a0 // double the input
addi $t1, $s1, -2 // subtract 2
beq $t1, $r0, skip // go to skip if it is zero
add $v0, $t1, $r0 // otherwise set the result to the double
jr $ra // return result
skip:
addi $v0, $r0, 2 // set it to 2 if it was zero
jr $ra // return result
```

`2015-Aug-ISA 2-3`

a. None
b. $s1
c. $s1 and $s0
d. $s1 and $t1

Compute needs to save any callee-saved registers it uses. In this case the only one is $s1.

53. The compute function will never be called by other code or from another place. What registers does the compute function need to save on the stack for it to work correctly?

```
begin:
add $t0, $t1, 3
ld $a0, 16($t0)
jal compute
add $s0, $t20, $t0
... (program ends eventually)
compute:
add $s1, $a0, $a0 // double the input
addi $t1, $s1, -2 // subtract 2
beq $t1, $r0, skip // go to skip if it is zero
add $v0, $t1, $r0 // otherwise set the result to the double
jr $ra // return result
skip:
addi $v0, $r0, 2 // set it to 2 if it was zero
jr $ra // return result
```

`2015-Aug-ISA 2-4`

a. None
b. $s1
c. $s1 and $s0
d. $s1 and $t1

Compute does not overwrite any registers that the rest of the program uses (it writes $s1, $t1, $v0 and the program has stored values in $t0 and $a0). Since no other code will ever call it, we don't need to save any registers for this to work correctly.

54. Does a procedure need to save $a and $v registers on the stack when it calls a sub-procedure?

`2015-Aug-ISA 2-5`

a. No, they are callee save
b. Depends on whether they are needed again after the sub-procedure returns
c. No, they are special registers
d. No, it is guaranteed to either not need them or overwrite them

If the procedure calls a sub-procedure before it has copied all of its argument values ($a registers) to other registers

then it would have to save them in case a sub-procedure overwrites them. If the procedure has written to an output register ($v) then it would need to save this as well in case the sub-procedure overwrites it.

55. This is the last code in a procedure. What instructions should be placed in _____?
lw $t0, 0($sp)
lw $t1, 4($sp)
_____
_____
2015-Aug-ISA 2-6
    a. addi $sp, $sp, 8; jr $ra
    b. addi $sp, $sp, -8; jal proceedure
    c. addi $sp, $sp, 2; jr $ra
    d. addi $sp, $sp, -2; jal proceedure

We are loading two words from the stack back into $t0 and $t1, so we need to move the stack pointer back UP by 2 words, or 8 bytes, and then do a return jump.

56. Why is it good to have consistent instruction encoding?
2014-Sep-ISA 2-1
    a. Smaller instructions
    b. Easier to handle constants
    c. Easier to decode instructions
    d. All of the above

Consistent instruction encoding makes it easier to decode the instructions because you handle each one the same way. However, it means that even very simple instructions need to have the same number of bits, so it can increase the average instruction size. It doesn't have any impact on the ability to handle constants.

57. A designer adds a "branch if register is zero" instruction to MIPS. (bzero REG, address) How far can this instruction jump if it uses a modified I-format encoding?
2014-Sep-ISA 2-2
    a. -2^{14} to 2^{14-1}
    b. -2^{16} to 2^{16-1}
    c. -2^{20} to 2^{20-1}
    d. -2^{21} to 2^{21-1}

The instruction only needs one register, so it has the I-format's 16 bits of immediate field plus the 5 bits from the 2nd register field in the I-format, for a total of 21 bits of immediate value. Since the branch is interpreted as instructions in 2's compliment, this is -2^20 to +2^20-1 instructions.

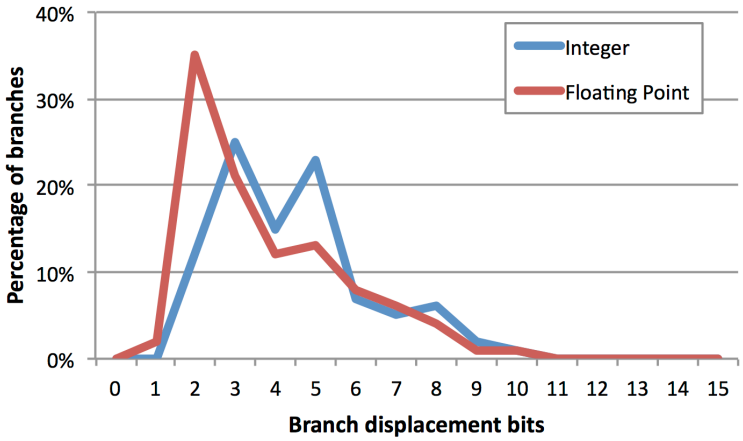
58. A designer adds a "load big constant to register" instruction to MIPS. (lbc REG, immediate) This instruction uses a modified I-format encoding. How much does this speed up loading a single 32-bit immediate?
2014-Sep-ISA 2-3
    a. No change
    b. 31% faster
    c. 65% faster
    d. 2x faster

The new instruction has 21 bits of immediate field (removing the second register gives you 16+5). However, you still need two instructions to load a 32 bit immediate, so it doesn't help.

59. Why does the MIPS ISA define a stack?
2014-Sep-ISA 2-4
    a. So procedures do not overwrite each other's registers and lose data
    b. So procedures can overwrite each other's registers without losing data
    c. Jump and link can't work without a stack
    d. All of the above

The stack allows programs to save register values that other procedures may need later so they can then overwrite those registers and restore the values when they are done. Jump and link works fine without a stack, but you would need to make sure you don't overwrite the RA register!

60. A designer comes up with a new instruction: branch-equal-constant that uses a modified I-encoding where the 2nd register is used to encode the branch address and the immediate field is used to encode the constant. What does this graph tell you about how useful this instruction would be? (floating point is the bar that is highest at 2.)



2014-Sep-ISA 2-5

a. No difference
b. Each constant branch will be twice as fast
c. ~85% of floating point and ~62% of integer code could use it to save 1 cycle for each branch
d. Can't tell because we don't know how many of the branches compare to constants

This new instruction will save one instruction from all branches that 1) compare a register to a constant and 2) need 5 bits or less of address. The graph above only shows us how many branches need 5 bits or less of address, but we do not know how many of those compare to a constant.

61. Does a procedure need to save $a1 and $v1 on the stack when it calls a sub-procedure?

2014-Sep-ISA 2-6

a. Yes, they are caller save
b. Depends on whether they are needed again after the sub-procedure returns
c. No, they are special registers
d. No, it is guaranteed to either not need them or overwrite them

If the procedure calls a sub-procedure before it has copied all of its argument values ($a registers) to other registers then it would have to save them in case a sub-procedure overwrites them. If the procedure has written to an output register ($v) then it would need to save this as well in case the sub-procedure overwrites it.

62. main() uses t0, t1, s0, B() uses t4, s3, s4, C() uses t1, t2, t3, t4, s0, s5. How many words on the stack are needed when main() calls B() and B() calls C()?

2016-Oct-ISA 2-1

a. 7
b. 8
c. 11
d. 12

main() is only a caller, so it saves: t0, t1, B is a caller and a callee, so it needs to save: t4, s3, s4, ra; and C is only a callee, so it saves: s0, s5 = 7 total

63. What happens to the current value of $RA when the JAL instruction is executed?

2016-Oct-ISA 2-2

a. Nothing
b. It gets overwritten
c. It gets stored on the stack
d. It is copied to $A0

JAL stores PC+4 into $RA when it executes.

64. Which of the following is true about this instruction?

```
addi RA, R31, -63000
```

2016-Oct-ISA 2-3

a. It is in a format that is not allowed by MIPS
b. It uses an immediate value that is now allowed by MIPS

c. It writes to a register that is not allowed by MIPS
d. It reads from a register that is not allowed by MIPS

The immediate values are two's complement 16-bit values, so they can go up to -2^15, which is -32768.

65. Which of these instructions have the shortest address range they can jump to?
2016-Oct-ISA 2-4
a. J
b. JR
c. JAL
d. BNE

BNE is an I-type instruction which means it can only use 16 bits to specify the offset.

66. Here is a small snippet of code that saves $t0 to the stack. What instruction should be put in the place of **X**?
x
```
sw $t0, 0($sp)
```
2016-Oct-ISA 2-5
a. addi $sp, $sp, 1
b. addi $sp, $sp, -1
c. addi $sp, $sp, 4
d. addi $sp, $sp, -4

We are storing a single word on the stack. Therefore, we need to move the stack pointer downwards by 4 bytes.

67. Does the ISA of a processor determine its microarchitecture?
2016-Oct-ISA 2-6
a. Yes. You have to build the hardware exactly the way the ISA specifies or programs won't run.
b. Yes. The ISA is the lowest-level description of the processor.
c. No. The compiler will fix the program when it is run if the processor has a different implementation
d. No. There are many ways to build a processor that has the same ISA.

Think about how we looked at single-cycle, multi-cycle, and pipelined implementations of the same ISA.

68. How is the immediate field interpreted for ANDI and ADDI instructions?
2016-Jan-ISA 2-1
a. ANDI=unsigned, ADDI=unsigned
b. ANDI=two's complement, ADDI=unsigned
c. ANDI=unsigned, ADDI=two's complement
d. ANDI=two's complement, ADDI=two's complement

ADDI interprets the immediate as two's compliment (it sign extends by replicating the MSB from the 16-bit immediate) while ANDI does not do this and interprets the value as unsigned.

69. Which of the following is true about this instruction?
```
addi RA, R32, -28000
```
2016-Jan-ISA 2-2
a. It is in a format that is not allowed by MIPS
b. It uses an immediate value that is not allowed by MIPS
c. It writes to a register that is not allowed by MIPS
d. It reads from a register that is not allowed by MIPS

This is a regular I-format instruction with a value safely within the two's complement 16bit range of the immediate field. Writing to RA is fine (it's just R31), but there is no R32 to read from, only R0-R31.

70. Which instruction can jump the furthest?
2016-Jan-ISA 2-3
a. J
b. JR
c. JAL
d. BNE

JR can use the full 32-bits of the register so it can jump further than the others which do not have a full 32-bits of immediate.

71. JAL and J are both J-format instructions. Which of the following statements is true?
`2016-Jan-ISA 2-4`
    a. JAL cannot jump as far as J
    b. JAL stores the next PC in the register file
    c. J stores a constant into the register file
    d. They can be used interchangeably

JAL and J can both jump equally far (since they are both J format and have the same size immediate) but JAL stores the next PC into the register file so that the program can return later with JR..

72. Why would it be a bad idea to create your own calling convention instead of using the standard MIPS convention?
`2016-Jan-ISA 2-5`
    a. You would overwrite stack/registers if you called other code that does use the standard convention
    b. The assembler would refuse to assemble your code because it does not meet the standard
    c. Your code would run more slowly because it would need to use more space on the stack to store registers
    d. B and C

You need to use the standard calling convention to ensure that your code works nicely with other code. This is essential for using libraries and other services (such as the operating system).

73. main() uses t0, t2, s0, B() uses t4, s3, s4, C() uses t1, t2, t3, t4, s0, s5. How many words on the stack are needed when main() calls B() and B() calls C()?
`2016-Jan-ISA 2-6`
    a. 8
    b. 9
    c. 11
    d. 17

main() is only a caller, so it saves: t0, t2, B is a caller and a callee, so it needs to save: t4, s3, s4, ra; and C is only a calle, so it saves: s0, s5 = 8 total.

74. Why do we only lose one register field for an I-format instruction vs. an R-format instruction?
`2013-Aug-ISA 2-1`
    a. Immediate values only need as many bits as a single register field
    b. The immediate value replaces other fields and not the register fields
    c. The immediate value replaces some of the register fields and some other fields
    d. The immediate instructions have more bits

Immediate instructions replace the rd, shmt, and funct fields. This adds up to 5+5+6=16 bits.

75. What do we do to the immediate field for branch instructions to calculate the branch address and why?
`2013-Aug-ISA 2-2`
    a. Shift to the left by 2 because instructions are word-aligned.
    b. Shift to the left by 4 because instructions are word-aligned.
    c. Add 4 because instructions are word-aligned.
    d. Sign-extend because the immediate is unsigned.

We shift the address to the left by 2 (multiply by 4) because instructions are word-aligned. We do add 4, but that's just because we always do PC+4, and we do sign-extend, but it is because the immediate is two's compliment.

76. How would you load the value 13 into the PC?
`2013-Aug-ISA 2-3`
    a. addi $ra, $r0, 13
       jr $ra
    b. addi $pc, $r0, 13
    c. lui $pc, 0
       ori $pc, 13
    d. lui $pc, 13

You can't write directly into the PC, but the jr instruction will load the value from register $ra into the PC.

77. What immediate value should the conditional branch at address 12 have to jump to address 8?
`2013-Aug-ISA 2-4`
    a. -1
    b. -2
    c. -4
    d. -8

-2. The next instruction will be PC+4+(imm<<2). 8=12+4+(-2*4)

78. Why do we have a convention for callee and caller saving of registers?
`2013-Aug-ISA 2-5`
    a. So we can call any function without overwriting our own registers or the function's registers.
    b. To avoid corrupting the stack.
    c. To make it possible for the compiler to generate code.
    d. All of the above.

The convention allows us to call other code without overwriting registers. Corrupting the stack simply requires incrementing and decrementing it correctly. The compiler would actually have an easier time if it didn't have to follow these rules.

79. A function uses registers $s0, $s1 and $t3. How much will it adjust the stack when it is called?
`2013-Aug-ISA 2-6`
    a. -1
    b. -2
    c. -4
    d. -8

The callee needs to save $s1 and $s0 on the stack, which is two words, or 8 bytes. So the stack needs to move down by -8.

80. If we added a new instruction format to MIPS that only specified one register, how large a constant could it hold?
`2013-Apr-ISA 2-1`
    a. 16 bits
    b. 17 bits
    c. 21 bits
    d. 32 bits

We would have the same as the I-format (16 bits) plus the space used for one register address, which is 5 bits, or a total of 21 bits.

81. What would happen if the ori instructions used a sign-extended value?
`2013-Apr-ISA 2-2`
    a. The upper 16 bits would sometimes be all 0s
    b. The upper 16 bits would sometimes be all 1s
    c. The lower 16 bits would sometimes be all 0s
    d. The lower 16 bits would sometimes be all 1s

The ori instruction ORs in the immediate into the lower 16 bits. If it was sign-extended, then if bit 15 was a 1, it would sign extend bits 16-31 to 1s, which when ORed in to the top 16 bits would result in all 1s in the top 16 bits.

82. Why do we only need a few bits for relative branches?
`2013-Apr-ISA 2-3`
    a. Branches are usually to nearby code
    b. Longer branches can be done with jump instructions
    c. The branch address is signed so we can jump forwards and backwards
    d. All of the above

All of these are correct. The key issue for performance is that most branches are to nearby code so we only rarely need all the bits in a jump instruction to get beyond the branch range.

83. main() calls procedure A(), which runs for a while and then calls procedure D(). A() uses registers s0, s1, s2, t0, t1, and t2. Which registers need to be saved right before A() calls D()?
`2013-Apr-ISA 2-4`
    a. ra, s0, s1, s2, t0, t1, t2
    b. s0, s1, s2, t0, t1, t2
    c. ra, t0, t1, t2
    d. t0, t1, t2

A() needs to save s0, s1, and s2 when it starts. Right before it calls D() it needs to save ra (so it knows where to return to) and t0, t1, and t2.

84. main() uses t0, t1, s0, s1, B() uses t4, s3, s4, C() uses t1, t2, t3, t4, s0, s5. How many words on the stack are needed when main() calls B() and B() calls C()?
`2013-Apr-ISA 2-5`
    a. 10
    b. 13
    c. 15
    d. 16

main() needs to save t0, t1 before it calls B(). B() needs to save t4 when it starts, and s3, s4 and ra when it calls C(). C() needs to save t1, t2, t3, and t4 before it starts. Main():2, B():4, C():4 = 10

85. What is an advantage of a memory-register machine over load-store-register machines?
`2013-Apr-ISA 2-6`
    a. Easier to build
    b. Simpler for the compiler
    c. Simpler assembly code
    d. Denser code

It is easier to write assembly code by hand if you have more complex instructions. Otherwise memory-register machines make everything more complicated and put many more addresses into the code.

3. Computer Arithmetic

1. What is the main reason we use two-level (1/0) digital logic instead of four-level (3/2/1/0) digital logic?
`2013-Dec-Computer Arithmetic-1`
    a. Noise
    b. Complexity
    c. Compatibility
    d. None of the above

Noise. Complexity is worth it if we can make the logic smaller; Compatibility is not an issue if each "new bit" holds exactly 2 "old bits".

2. Why can't we just look at the carry out in a two's compliment number to see if we have an overflow?
`2013-Dec-Computer Arithmetic-2`
    a. We can
    b. Carry out means the number is negative
    c. Negative numbers have all 1s to the left of the MSB
    d. Negative two's compliment numbers can't overflow in addition

Negative two's compliment numbers have an infinite number of 1s to the left. (This is what sign extension does.) So when the carry out is 1 it may just mean that the number is negative.

3. What is so nice about two's compliment numbers?
`2013-Dec-Computer Arithmetic-3`
    a. Positive and negative zeros
    b. Subtraction uses a regular adder
    c. Same range for positive and negative numbers
    d. All of the above

Subtraction uses a regular adder: A-B = A+!B+1. Two's compliment has 1 more negative number than positive

number for a given number of bits, but only has one zero.

4. The number 0110 cannot represent which of the following in fixed point:
   2013-Dec-Computer Arithmetic-4
   a. 3
   b. 0.75
   c. 0.625
   d. 0.375

   0.625 = 0.101

5. What is the result of adding 1000 and 0000 to a 4-bit signed-magnitude number?
   2013-Dec-Computer Arithmetic-5
   a. Depends on the number you're adding to
   b. Negative
   c. Overflow
   d. No change

   1000 and 0000 are -0 and +0 in signed-magnitude representation, so adding them to a number does not change it.

6. The exponent in a floating point number does what?
   2013-Dec-Computer Arithmetic-6
   a. Shifts the binary point
   b. Multiplies the fraction
   c. Normalizes the number
   d. Implies a 1 before the fraction

   The exponent controls how much the binary point is shifted by multiplying the fraction by 2^exponent. This allows a positive exponent to shift the binary point to the right and a negative one to the left.

7. What would the following 2's complement math compute: (!A+1) + (!B + C)
   2017-Jan-Computer Arithmetic-1
   a. A-C+B
   b. C-A-B-1
   c. C-A-B
   d. C-A+B

   To subtract in 2's complement you need to do X + !Y +1. So (!A+1)+C = C-A. Then C-A +(!B) = (C-A) -B -1 =C-A-B-1

8. Why do computers use binary signals (1 and 0 only) instead of analog signals (including fractional values)?
   2017-Jan-Computer Arithmetic-2
   a. We cannot build math circuits for analog signals
   b. Analog signals change more slowly
   c. Binary signals are more immune to noise
   d. More than one of the above

   We can absolutely build (very fast) circuits that do math on analog signals. Indeed we can build faster circuits in some cases than the digital equivalents. Analog signals can change very fast (think of the radio signals going to a mobile device) but analog signals pick up noise and it is hard to remove. Digital circuits are always a 1 or 0 so as long as it is close to one or the other we can easily remove the noise.

9. What is the largest magnitude number you can represent with a two's complement fixed-point ddd.dd format?
   2017-Jan-Computer Arithmetic-3
   a. -4
   b. -3.75
   c. 3.75
   d. 4

   100.00 will be -4. The largest positive number would be 011.11 which would be 3.75.

10. The number 0110 cannot represent which of the following in fixed point:

2017-Jan-Computer Arithmetic-4

a. 0.375
b. 0.625
c. 0.750
d. 3.000

0.625 (base-10) = 0.101 (base-2)

11. Which of the following statements is true?

2017-Jan-Computer Arithmetic-5

a. A non?normalized floating?point number can represent the same number in several ways
b. Signed?magnitude has a wider range than two's complement numbers
c. To get the absolute value of a two's complement number, you remove MSB and negate the remaining bits
d. You can detect arithmetic overflow on unsigned numbers by comparing carry in and carry out of the MSB

A, but not if the floating point number is normalized.

12. What is the range of an 8?bit signed magnitude binary number?

2017-Jan-Computer Arithmetic-6

a. ?64 to 64
b. ?64 to 63
c. ?128 to 127
d. -127 to 127

1 bit for the sign, and 7 bits for magnitude (0 ... 27 - 1=0 ... 127)

13. What is the value of the binary string 1101?

2014-Apr-Computer Arithmetic-1

a. -5
b. -3
c. 13
d. Could be any of the above

The value depends on how you interpret the 1s and 0s. Without being told how to interpret it you can't tell.

14. Why do we use binary numbers in computers?

2014-Apr-Computer Arithmetic-2

a. They are faster than base 10 signals
b. They take up less space than base 10 signals
c. They are more immune to noise than base 10 signals
d. All of the above

The key reason is that they are more immune to noise than trying to store multiple values on each wire. The individual binary arithmetic might be faster than base 10, but you also have more bits to process. They certainly don't take up less space than being able to store the values 0-9 in a single bit!

15. Which 8-bit unsigned fixed-point representation would be most appropriate for storing the speed of a car whose speedometer goes up to 200km/h?

2014-Apr-Computer Arithmetic-3

a. 8.0
b. 7.1
c. 6.2
d. 5.3

8.0 8 bits gets you up to 2^8 which is 256. Since the speedometer can go up to 200km/h we need to get that high. We won't be able to tell the difference between 10 and 10.5km/h, though, since the smallest value will then be 1km/h

16. Why can we get unexpected results from floating-point math when we do (big+small)-big?

2014-Apr-Computer Arithmetic-4

a. Floating point addition is inaccurate regardless of the size of the numbers
b. Combining large and small numbers can lose precision
c. Both A and B

d. Neither

When we combine large and small numbers we have to adjust one of them to have the same exponent as the other, which may result in losing its value. Floating point addition can be perfectly accurate if the numbers are of the same magnitude so no shifting is required.

17. A serial multiplier trades off what for what vs. a parallel multiplier? (The serial multiplier is...)
`2014-Apr-Computer Arithmetic-5`
    a. Lower precision, but faster
    b. Faster, but larger circuit area
    c. Slower, but smaller circuit area
    d. They are the same, just different ways to look at the circuit

The serial multiplier does each step one after another and therefore only needs one adder and one shifter. The parallel multiplier does all the shifts and adds in parallel, which requires much more circuit area, but operates much faster.

18. When a floating-point number is normalized by shifting the mantissa to the left, what happens to the exponent?
`2014-Apr-Computer Arithmetic-6`
    a. Increases
    b. Decreases
    c. Stays the same
    d. Depends on the number

Every time you shift the mantissa to the left you are multiplying it by 2, so the exponent has to go down by 1.

19. Will the results be correct if you use an ALU designed for 32-bit two's complement addition for 32-bit unsigned integers?
`2015-Oct-Computer Arithmetic-1`
    a. Yes, however the overflow flag is calculated differently, and may be incorrect
    b. No, if MSB is 1 for any of the inputs, then the answer will be incorrect
    c. Yes, the same algorithm is used, regardless whether the input happen to be signed or not
    d. No, all signed numbers are subjected to the operation NOT(input) + 1 before an addition, which would produce garbage in the unsigned case

Two's compliment uses the same logic as unsigned except for overflow.

20. Is it possible to complete hardware binary multiplication of two 32 bit numbers in a single clock cycle?
`2015-Oct-Computer Arithmetic-2`
    a. Yes
    b. No, it would require too much power
    c. No, but it will likely be possible in the future, as chips get smaller
    d. No, you need registers to store partial products, which requires extra cycles

Yes it is possible. However the propagation delay would be long and it would require a fair amount of chip area, compared to an iterative solution.

21. For a 7-bit normalized floating-point format: (-1)^S*(.FFFF)*(2^EE) where FFFF is unsigned and EE is two's complement, what is the smallest positive value you can represent?
`2015-Oct-Computer Arithmetic-3`
    a. 0.5
    b. 0.25
    c. 0.125
    d. 0.28125

(0 0000 10) = 1*1.0*2-2 (=1/4).

22. Compute 1010 - 1110 for 4-bit two's complement numbers?
`2015-Oct-Computer Arithmetic-4`
    a. Overflow
    b. +4
    c. +7
    d. -4

1010 - 1110 is -6- -2 = -4, which can be represented as 1100.

23. What is the range of an 8-bit signed magnitude binary number?
    2015-Oct-Computer Arithmetic-5
    a. -64 to 64
    b. -64 to 63
    c. -128 to 127
    d. -127 to 127

    1 bit for the sign, and 7 bits for magnitude (0 ... 27 - 1=0 ... 127)

24. Which of the following statements is true?
    2015-Oct-Computer Arithmetic-6
    a. A non-normalized floating point number can represent the same number in several ways
    b. Signed magnitude has a wider range than two's complement numbers
    c. To get the absolute value of a two's complement number, you remove MSB and negate the remaining bits
    d. You can detect arithmetic overflow on unsigned numbers by comparing carry in and carry out of the MSB

    A, but not if the floating point number is normalized.

25. When adding a positive and a negative two's complement number, can you cause an overflow?
    2018-Oct-Computer Arithmetic-1
    a. No, because addition of two's complement values cannot an overflow
    b. No, because addition of opposite signed two's complement values cannot cause an overflow
    c. Yes, if the positive number is large enough
    d. Yes, if the negative number is large enough

    When adding two numbers of opposite signs in two's complement you never get an overflow as the final value must be less than the maximum positive value and greater than the minimum negative value.

26. What is the result of (-4)+(-4) for two 4-bit two's complement numbers?
    2018-Oct-Computer Arithmetic-2
    a. 1000 (no overflow)
    b. 0000 (overflow)
    c. 0000 (no overflow)
    d. 1111 (overflow)

    A 4-bit two's complement number has the range of 1000=-8 to 0111=+7. So -4 + -4 will give us -8.

27. Why do we use 2's complement numbers in computers?
    2016-Aug-Computer Arithmetic-1
    a. Subtraction uses a regular adder
    b. Same range for positive and negative numbers
    c. Positive and negative zeros
    d. All of the above

    Subtraction uses a regular adder: A-B = A+!B+1. Two's compliment has 1 more negative number than positive number for a given number of bits, but only has one zero..

28. Base-2 arithmetic is what is used in our MIPS processor: numbers are stored in binary (1s and 0s) and each binary digit represents twice the value before it. Another approach is to use Base-10 number representations, where we use 4 binary digits together to represent the numbers 0-9. (Representing 0-99 would require two sets of 4 bits, one for the one's digit and one for the ten's digit, and representing 0.0 to 9.9 would require the same number of bits.) We also need to build Base-10 adders that can do the right thing with this representation. Why would one want to use base-10 arithmetic in computers instead of base-2 arithmetic? (Hint: IBM actually makes processors that have hardware for Base-10. Why would they do that?)
    2016-Aug-Computer Arithmetic-2
    a. It is easier for humans to reason about because we think in Base-10
    b. It is easier for computers to work with because carries are simpler in adders
    c. It is easier to avoid rounding errors with values like money that are already in Base-10
    d. None of the above

Base-2 arithmetic has rounding rules that look different from base-10 rounding rules. Typically, money and other human inputted data is represented in base-10 which would require extensive checking and overhead if one were to try and emulate the results with base-2 arithmetic.

29. The number 0110 cannot represent which of the following in fixed point:
`2016-Aug-Computer Arithmetic-3`
    a. 0.375
    b. 0.625
    c. 0.750
    d. 3.000

0.625 (base-10) = 0.101 (base-2)

30. What are the benefits of fixed-point over floating-point arithmetic?
`2016-Aug-Computer Arithmetic-4`
    a. Subtraction uses only a regular adder without input or output modifications
    b. Multiplication uses only a regular multiplier without input or output modifications
    c. Both a. and b.
    d. Neither a. nor b.

Subtraction required the bits to be modified on input, and to provide a carry in. Multiplication of fixed-point numbers requires shifting of the output to preserve the binary-point.

31. What are the benefits of denormal numbers? (That is numbers that do not require the floating point number to have a 1. first.)
`2016-Aug-Computer Arithmetic-5`
    a. It allows for a very efficient number format to compute results
    b. It is an inefficient format, but provides more rounding modes
    c. It allows for a larger range of numbers at the cost of precision
    d. None of the above

Normal numbers provide for a uniform way to represent numbers in floating point arithmetic, so that there is only one way to represent a number. Denormal numbers are numbers that do not hold to the normalization guidelines (it allows zeros in the front of the significand). The result is an expanded range of numbers that allow for access to absolutely smaller numbers with less precision.

32. When does floating-point addition take more logic to compute than fixed-point addition?
`2016-Aug-Computer Arithmetic-6`
    a. Sometimes
    b. Always
    c. Never
    d. Unable to determine

Floating point addition needs to handle alignment of the binary point (and tends to handle overflow, rounding conditions and denormal numbers (depending on the specification)). This will require additional logic.

33. Convert the following 5-bit sign/magnitude binary number to a 5-bit two's complement number: 11111
`2014-Dec-Computer Arithmetic-1`
    a. 01111
    b. 10000
    c. 10001
    d. Can't be represented as a 5-bit two's complement number

11111 = - 1111 = -15. In two's complement, 10000 = -16, so we need to add 1, and get 10001=-15.

34. What would the following 2's complement math compute: (A+1) + (!B + !C)
`2014-Dec-Computer Arithmetic-2`
    a. A-B-C-1
    b. A-B+C-1
    c. A+B+C-1
    d. A-B+C

To subtract in 2's complement you need to do A + !B +1. So (A+1)+!B = A-B. Then A-B + !C = (A-B + !C +1) -1 = A-B-C-1

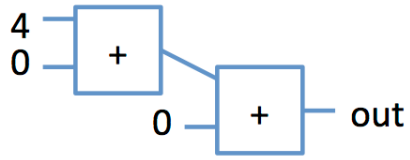
35. What do we have to do to convert the decimal number 3 to two's complement? (Assume there are enough bits to represent the number.)

2014-Dec-Computer Arithmetic-3

a. Convert to unsigned binary
b. Convert to unsigned binary and invert the most significant bit
c. Convert to unsigned binary and invert all the bits
d. Convert to unsigned binary, invert all the bits, and add one

Since 3 is a positive number, the two's complement representation is the same as the unsigned binary, as long as it fits into the number of bits available.

36. A designer decides to build an analog computer and uses voltages 0-10V rounded in 1V increments for signals (e.g. 2V = the decimal number 2.) The designer builds an adder that works pretty well, but has +10%V of noise on the output. The designer connects 4V and 0V to one adder, and then uses a second adder to add that result to 0V. What is the output of the circuit as built, and what is the desired output of the circuit?



2014-Dec-Computer Arithmetic-4

a. Output=4, desired=5
b. Output=4, desired=4
c. Output=5, desired=5
d. Output=5, desired=4

The first adder does 0+4 but gives 4.4 as the answer since it has +10% noise on the output. The second adder does 0+4.4 and gives 4.84 due its noise, which is closest to 5. The desired result is 0+0+4=4.

37. Can you tell which of these numbers is closer to zero? They are both using the same format, but they may be two's compliment, signed-magnitude, fixed-point, or seeeffff floating point (-1^Sx0.ffffx2^(eee-4)).
A=01110001
B=01010011

2014-Dec-Computer Arithmetic-5

a. A
b. B
c. Depends on where the binary point is
d. Depends on whether they are two's complement

All of these formats allow you to directly compare non-negative numbers for magnitude. Even the floating point one lets you do this since the exponent comes first. Therefore we can conclude that B is smaller since it doesn't have a 1 in the 5th digit.

38. Without normalization, how many ways are there to represent the number 3 with an seeeffff floating point format? (-1^Sx0.ffffx2^(eee-4))

2014-Dec-Computer Arithmetic-6

a. 1
b. 2
c. 3
d. 4

We can have ffff = 0011, 0110, or 1100, but 0011 requires a shift of 4 (11.0000) which is more than eee-4 (max 111-4=7-4=3) can support. This means we can only represent 3 with 0110 (shifted 3) or 1100 (shifted 2).

39. If we changed our 32-bit processor to have 256 registers, have the same size instructions, and keep similar I-, R-, and J-formats, how many bits would we waste if we loaded a 32-bit constant?

2015-Apr-Computer Arithmetic-1

a. 0
b. 2
c. 4

d. 8

With 256 registers we need 8 bits per register. This means the I-type instructions will need 16 bits for the register values, which is 6 more than the 10 bits they have with 32 registers. This would reduce the size of their immediate field from 16 bits to 10 bits, which would require 4 immediate constants to fill up a 32 bit constant. With 4 10-bit immediates we would load 40 bits, which means we would waste 8 bits for our 32 bit constant.?

40. Which of the following statements is not true about two's compliment notation?
`2015-Apr-Computer Arithmetic-2`
    a. We can easily determine if the number is negative by looking at the MSB
    b. Addition is a non-inverted addition with carry set to 1
    c. Subtraction inverts one input and does an addition with carry in set to 1
    d. There is only one zero

To subtract we need to set carry to 1 and add the inverse of the number we want to subtract.

41. For an 8-bit floating-point format: (-1)^S*(.FFFF)*(2^EEE) where FFFF is unsigned and EEE is two's complement, what is the largest positive value you can represent?
`2015-Apr-Computer Arithmetic-3`
    a. 125
    b. 31.75
    c. 15.875
    d. 7.5

The largest number is given by (-1)^0*(.1111)*(2^011) = .1111<<3 = 111.1 = 7.5

42. Which of these is not a valid interpretation of the bits 1010 according to the representations we've discussed?
`2015-Apr-Computer Arithmetic-4`
    a. -6
    b. -2
    c. 2.75
    d. 10

Two's compliment: -6, Signed-magnitude: -2, Unsigned: 10, 2.2 unsigned: 2.5,

43. How are binary numbers distributed along the number line for the fixed-point number format?
`2015-Apr-Computer Arithmetic-5`
    a. Linearly
    b. Exponentially
    c. Depends on the binary point location
    d. Depends on the exponent

Fixed-point counts just like regular binary, but everything is divided by the binary point shift. E.g., 2.2 counts 0,1,2,3, etc, but everything is divided by 4 (shifted right by two decimal places) so the interpretation is 0.0, 0.25, 0.5, 0.75, etc. Only floating point has exponentially distributed numbers.

44. Which is not a benefit of floating point over fixed-point?
`2015-Apr-Computer Arithmetic-6`
    a. Floating point can represent larger numbers
    b. Floating point can represent smaller numbers
    c. Floating point addition is simpler
    d. Floating point does not require more bits

Addition is much more complex because you have to shift to make the exponents match first. You don't need more bits to do floating point; you just use them differently.

45. Compute 0011 - 1011 for 4-bit signed magnitude numbers.
`2013-Jan-Computer Arithmetic-1`
    a. 0000
    b. 0110
    c. 1000
    d. 1110

46. Compute 1101 - 1011 for 4-bit two's complement numbers.
`2013-Jan-Computer Arithmetic-2`
    a. 0001
    b. 0010
    c. 1000
    d. overflow

47. For an 8-bit, signed-magnitude, fixed-point number format #####.###, what is the largest positive value you can represent?
`2013-Jan-Computer Arithmetic-3`
    a. 7.5
    b. 15.875
    c. 31.875
    d. 120

48. For an 8-bit floating-point format: (-1)^S*(.FFFF)*(2^EEE) where FFFF is unsigned and EEE is two's complement, what is the largest positive value you can represent?
`2013-Jan-Computer Arithmetic-4`
    a. 7.5
    b. 15.875
    c. 31.875
    d. 120

49. How are values distributed along the number line for the fixed-point format?
`2013-Jan-Computer Arithmetic-5`
    a. Exponentially
    b. Linearly
    c. Depends on the exponent
    d. None of the above

50. What is the benefit of floating point over fixed point?
`2013-Jan-Computer Arithmetic-6`
    a. Can represent smaller numbers
    b. Can represent larger numbers
    c. Can provide more precision for small numbers or range for large numbers
    d. Simplifies addition circuits

51. Why do computers use binary signals (1 and 0 only) instead of analog signals (including fractional values)?
`2015-Aug-Computer Arithmetic-1`
    a. We cannot build math circuits for analog signals
    b. Analog signals change more slowly
    c. Binary signals are more immune to noise
    d. More than one of the above

    We can absolutely build (very fast) circuits that do math on analog signals. Indeed we can build faster circuits in some cases than the digital equivalents. Analog signals can change very fast (think of the radio signals going to a mobile device) but analog signals pick up noise and it is hard to remove. Digital circuits are always a 1 or 0 so as long as it is close to one or the other we can easily remove the noise.

52. What is the result of the following signed-magnitude addition: 1000+0001?
`2015-Aug-Computer Arithmetic-2`
    a. -1
    b. 0
    c. 1
    d. 8

    This is -0 + 1 which is 1.

53. What would the following 2's complement math compute: (!A+1) + (!B + C)
`2015-Aug-Computer Arithmetic-3`
    a. A-C+B
    b. C-A-B-1
    c. C-A-B
    d. C-A+B

To subtract in 2's complement you need to do X + !Y +1. So (!A+1)+C = C-A. Then C-A +(!B) = (C-A) -B -1 =C-A-B-1

54. Which of the following number formats has only one zero?
`2015-Aug-Computer Arithmetic-4`
    a. Signed magnitude
    b. Two's Complement
    c. Floating Point
    d. More than one of the above

Signed magnitude and floating point both have negative and positive zeros. Two's complement has only one zero..

55. How many bits do you need to store the temperature in Sweden to an accuracy of 0.2 degrees, assuming it ranges from -40 to +20?
`2015-Aug-Computer Arithmetic-5`
    a. 6.3 bits
    b. 7 bits
    c. 9 bits
    d. You can't perfectly represent 0.2 in binary

-40 to +20 is a range of 60 and we want it to an accuracy of 0.2, so that means we need 60*5=300 numbers. So with 9 bits and a number format where we interpret it as 0 starting at -40 and each increment going up by 0.2 we can store the whole range.

56. What is the largest magnitude number you can represent with a two's complement fixed-point ddd.dd format?
`2015-Aug-Computer Arithmetic-6`
    a. -4
    b. -3.75
    c. 3.75
    d. 4

100.00 will be -4. The largest positive number would be 011.11 which would be 3.75.

57. What value can the binary number 10011 represent?
`2014-Sep-Computer Arithmetic-1`
    a. 19
    b. -3
    c. -13
    d. All of the above

10011 is 19 if it is unsigned, -3 if it is sign-magnitude, and -13 if it is 2's compliment. The value of a binary string depends on the interpretation!

58. A company develops two new types of transistors: LF transistors are very large, but very fast, and SS transistors are very small but slow. How would you design a multiplier for each type of transistor to match its characteristics?
`2014-Sep-Computer Arithmetic-2`
    a. LF: serial, SS: serial
    b. LF: serial, SS: parallel
    c. LF: parallel, SS: parallel
    d. LF: parallel, SS: serial

For LF you want a small circuit, but it is okay if it takes many cycles since they are fast transistors: serial. For SS you can have a large circuit (since they are small) but it shouldn't take many cycles since they are slow: parallel.

59. How many bits do you need to store the weight in a scale that can measure up to 2kg with an accuracy of 0.01kg?
`2014-Sep-Computer Arithmetic-3`
    a. 11 bits
    b. 2.7 bits
    c. 8 bits
    d. You can't represent 0.01 perfectly with binary numbers

To cover the range of 2kg to 0.01kg you have 200 numbers (200*0.01=2). To have 200 numbers you need 8 bits since

2^8=256. If the scale then treats the count as counts of 0.01kg, you can exactly represent weights from 0 to 2.55kg with an accuracy of 0.01kg.

60. What would the following 2's compliment math compute: A+!B
`2014-Sep-Computer Arithmetic-4`
    a. A-B
    b. A+B-1
    c. A-B-1
    d. Garbage

To subtract in 2's compliment you need to do A+!B+1, so A+!B would compute A-B-1.

61. Why do you shift the mantissa in floating point addition?
`2014-Sep-Computer Arithmetic-5`
    a. To make the exponents match
    b. To line up the binary points
    c. To make the mantissas have the same magnitude so you can add them
    d. All of the above

To add two floating point numbers they need to be the same magnitude. Shifting the mantissa does this by making the exponents match, which is equivalent to lining up the binary points.

62. Why do computers use binary numbers instead of tertiary (3 levels) or analog (infinite levels) signals?
`2014-Sep-Computer Arithmetic-6`
    a. Can't design non-binary circuits
    b. Can integrate lots of circuits without noise adding up
    c. A and B
    d. None of the above

We can design non-binary circuits (think analog OpAmps) but the reason for binary numbers is that they are much more immune to noise which makes it possible to integrate lots of circuits and have them work.

63. You want to compute a two's complement add, but accidentally use an unsigned add instruction instead. What can you say about the result, compare to a signed add?
`2016-Oct-Computer Arithmetic-1`
    a. Signed and unsigned add is the same in hardware, result is always the same
    b. The output is the same only if the input values are positive
    c. The output is always the same, but the overflow flag may be incorrect
    d. It reduces the instruction latency

Two's complement numbers use a regular adder, but the detection of overflow is different.

64. For a 6-bit normalized floating-point format: (-1)^S*(.FFF)*(2^EE) where FFFF is unsigned and EE is two's complement, what is the smallest positive value you can represent?
`2016-Oct-Computer Arithmetic-2`
    a. 0.5
    b. 0.25
    c. 0.125
    d. 0.28125

0 000 01 = 1*1.0*2^(-2) = 0.25

65. Compute 1010 - 1101 for 4-bit two's complement numbers.
`2016-Oct-Computer Arithmetic-3`
    a. Overflow
    b. 3
    c. -3
    d. -4

1010 - 1101 = -6 -(-3) = -3, which can be represented as 1101

66. What is the range of a 3-bit signed magnitude binary number?

2016-Oct-Computer Arithmetic-4

a. -3 to 3
b. -4 to 3
c. -7 to 7
d. -7 to 8

Maximum is 011 = 3 and minimum is 111 = -3.

67. Does binary addition take more logic to compute than floating-point addition for the same number of bits?

2016-Oct-Computer Arithmetic-5

a. Sometimes
b. Always
c. Never
d. Unable to determine

Floating point addition needs to do a shift to match the exponents to do the addition..

68. What is the main reason we do not use quaternary (4-level, 0/1/2/3) logic instead of binary logic?

2016-Oct-Computer Arithmetic-6

a. Noise
b. Complexity
c. Compatibility
d. It is impossible to build in hardware

It is harder to keep the circuits from switching to the wrong values with random noise.

69. What is the following 5-bit sign-magnitude number in 5-bit two's complement: 01111?

2016-Jan-Computer Arithmetic-1

a. 10000
b. 01111
c. 11111
d. It cannot be represented in 5-bit two's complement

01111 is 15 in sign-magnitude (the first bit is 0, so it is positive). A 5-bit two's complement number can represent 15 as its largest number in the same way: 01111.

70. A 4-bit adder was built with a defect that results in the lowest input bit (LSB) for one of the inputs always being stuck at 1. What is the largest absolute error that may result from this defect when adding unsigned numbers?

2016-Jan-Computer Arithmetic-2

a. 0
b. 1
c. 15
d. 16

If you added 0000+1111 you would expect 1111=15, but if the adder set the first input to be 0001 you would get 0001+1111=0000=0 instead (overflow). This is an error of 15.

71. What is the largest positive value you can represent with a 6-bit (-1)^S*(0.FF)*(2^EEE) floating point format, where FF is unsigned and EEE is two's complement?

2016-Jan-Computer Arithmetic-3

a. 6
b. 14
c. 96
d. 224

011011=0.11*2^.011 = 0.11*2^3=110 = 6

72. Which of the following are true about which numbers can be represented with 4-bit two's complement but not with 4-bit sign-magnitude?

2016-Jan-Computer Arithmetic-4

a. 0 can be represented with 4-bit two's complement but not 4-bit sign-magnitude
b. -7 can be represented with 4-bit two's complement but not 4-bit sign-magnitude

c. All numbers that can be represented with 4-bit two's complement can also be represented with 4-bit sign-magnitude
d. All numbers that can be represented with 4-bit sign-magnitude can also be represented with 4-bit two's complement

Two's complement can represent -(2^(N-1)) to 2^(N-1)-1. Signed magnitude can represent -(2^(N-1)-1) to (2^(N-1)-1).

73. How can you calculate the two's complement subtraction 1100-1010 with regular adders?
`2016-Jan-Computer Arithmetic-5`
a. 1100+1010
b. 1100+0101
c. 1100+0101+1
d. None of the above

Two's complement is so useful because A-B can be done as A+!B+1.

74. What is the most negative value you can represent with a 5-bit, 3.2 sign-magnitude fixed-point number?
`2016-Jan-Computer Arithmetic-6`
a. -0.125
b. -0.25
c. -3.75
d. -7.75

To be negative the first bit needs to be 1. To be large we want all bits to be 1. So we have 111.11 = -11.11=-3.75.

75. Why do we use two's compliment notation?
`2013-Aug-Computer Arithmetic-1`
a. There is only one zero
b. Subtraction is just invert addition with the carry in set to 1
c. We can easily determine if the number is negative by looking at the MSB
d. All of the above

All of the above.

76. How many bits are required to compute the following in two's compliment without an overflow? -8+8=0
`2013-Aug-Computer Arithmetic-2`
a. 4
b. 5
c. 6
d. 7

To hold the values 8 and -8 we need 4 bits for 8 and 5 bits for -8. Since the result is 0, we don't need more bits for the result. Therefore we need 5 bits and get: -8+8=11000+01000 = 00000.

77. What number format gives you the largest range for a 3-bit fixed-point number?
`2013-Aug-Computer Arithmetic-3`
a. xxx.
b. xx.x
c. x.xx
d. .xxx

xxx. gives you a range of 000 to 111 = 0 to 7.

78. Why did processors switch from serial multipliers to parallel multipliers?
`2013-Aug-Computer Arithmetic-4`
a. Parallel multipliers became easier to design
b. We had enough transistors that we didn't care about size
c. Serial multipliers requires less logic
d. All of the above

Parallel multipliers use a huge amount more logic, but as transistor sizes have shrunk we have tons of transistors to use so this is much less of an issue.

79. Which of these floating point operations is most likely to cause a divide by zero exception if there aren't enough bits in the number representation?
2013-Aug-Computer Arithmetic-5
    a. 1/(($2.1x10^{32}$+$5.6x10^{2}$)-$2.1x10^{32}$)
    b. 1/(($5.6x10^{2}$+$2.1x10^{2}$)-$5.6x10^{2}$)
    c. 1/(($2.1x10^{32}$-$2.1x10^{32}$)-$5.6x10^{2}$)
    d. 1/(($5.6x10^{2}$-$2.1x10^{2}$)-$5.6x10^{2}$)

    (big+small) = big if we don't have enough bits to shift the small value over to match the exponents. In that case we will have (big+small)-big = 0, and 1/0.

80. Floating point normalization solves what problem?
2013-Aug-Computer Arithmetic-6
    a. Exponential spacing of the values on the number line
    b. Linear spacing of the values on the number line
    c. Multiple representations for the same value
    d. Needing to shift the exponent for addition

    Normalization forces a single representation for each value by insisting that the mantissa be shifted to the left as far as possible. The implied 1 before the mantissa in the IEEE standard forces the mantissa value to be shifted left until there is a leading 1.

81. How many additions are needed in a serial multiplier to multiply 1101 by 1000?
2013-Apr-Computer Arithmetic-1
    a. 1
    b. 3
    c. 4
    d. 16

    We only need 1 because we have 1 one and 3 zeros in the multiplicand.

82. What is the biggest problem with signed magnitude numbers?
2013-Apr-Computer Arithmetic-2
    a. Subtraction is messy
    b. We have two zeros
    c. We can't detect overflow
    d. We use a bit for the sign

    Two zeros is a waste, but not a terrible one. The biggest problem is the hassle of subtraction where you have to look at the sign and magnitude separately.

83. What is the result of (-4)+(-6) for 4 bit two's complement numbers?
2013-Apr-Computer Arithmetic-3
    a. 10111
    b. 10110
    c. 0111
    d. Overflow

    -4 + -6 is -10, which can not be represented with a 4-bit two's complement number. (minimum value is -8).

84. What are the largest and smallest values you can represent with a 5-bit unsigned fixed-point number?
2013-Apr-Computer Arithmetic-4
    a. 31 to 0.3125
    b. 31 to 1
    c. 0.96875 to 0.03125
    d. 15.75 to 0.25

    11111=31 .00001= 0.03125 The problem did not specify where you put the binary point, so you can move it to get the full range.

85. Why do we use normalized numbers in floating point?
2013-Apr-Computer Arithmetic-5

a. To force only one representation of each number
b. To provide an exponential scale
c. To force the mantissa to start with a 1
d. To avoid negative mantissas

Normalization means that we interpret all mantissas as 1.mmmm, so there is only one way to represent each number since we are forced to have 1.000 and not 0.100x2, 0.010x4, etc.

86. What do you expect the result to be of the following in a floating point format with a less than 28 bits for the mantissa? (2x1030+5x102)-2x1030
`2013-Apr-Computer Arithmetic-6`
    a. 0
    b. 2x1030
    c. 5x102
    d. -2x1030

We have (big+small)-big, and we know that for a limited precision number (limited number of mantissa bits) big+small=big, so we should get big-big=zero.

4. Logic

1. How did we use a decoder in the SRAM example?
`2013-Dec-Logic-1`
    a. To split the address for the rows and columns
    b. To select the correct columns
    c. To select the correct row
    d. To convert from one-hot input to binary output

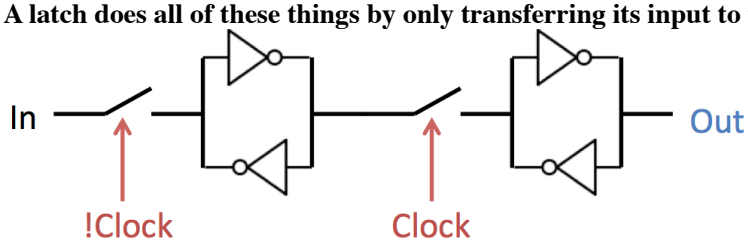
The decoder took in part of the binary address and converted it to a one-hot signal to select the correct row to activate.

2. Is it easier to convert from an optimized circuit to a truth table or from a truth table to an optimized circuit?
`2013-Dec-Logic-2`
    a. Easier to go from an optimized circuit to truth table
    b. Easier to go from a truth table to an optimized circuit
    c. Both the same
    d. Trick question: if you can use a Karnaugh map then it is the same, otherwise it is harder to convert to a circuit

It is generally harder to convert form a truth table to an optimized circuit because there are so many possible circuits for each truth table. Karnaugh maps are one way of helping, but they are not less complicated than brute-forcing a truth table.

3. What does a latch do?
`2013-Dec-Logic-3`
    a. Store a value
    b. Update the stored value on the clock signal
    c. Break the feedback loop in a sequential circuit
    d. All of the above

A latch does all of these things by only transferring its input to the output when the clock changes.



4. What would change if we reversed Clock and !Clock in the FlipFlop above?
`2013-Dec-Logic-4`
    a. Nothing
    b. The output would be inverted

c. The latch would be "transparent" and the data would go right through
d. The latch would be updated when the clock goes from 1 to 0

Switching Clock and !Clock would cause the data to be updated in the FlipFlop when the clock goes from 1 to 0 (falling edge) rather than 0 to 1.

5. How many output bits does an adder have if it is adding two 4-bit numbers?
`2013-Dec-Logic-5`
   a. 4
   b. 5
   c. 6
   d. Depends on whether the numbers are two's compliment or fixed-point

4 bits plus 1 carry out, for 5 bits. This is the same for twos compliment.

6. Why are SRAMs faster than DRAMs?
`2013-Dec-Logic-6`
   a. SRAMs use powered inverters to store data
   b. SRAMs use un-powered capacitors to stores data
   c. SRAMs have longer wires connecting all the bits
   d. They aren't: SRAMs are slower than DRAMs

DRAMs store data in capacitors. The capacitors only have a small amount of charge, and this charge needs to go out to the output wire to change its voltage. SRAMs have powered inverters so they can get charge from the power supply to change the voltage on the output lines.

7. How many inputs can a 6-bit multiplexer with 4 control bits switch between?
`2017-Jan-Logic-1`
   a. 4
   b. 6
   c. 16
   d. 64

4 control bits means the MUX can choose between 24=16 possible inputs. Each input has 6 wires.

8. Why are DRAMs so much cheaper than SRAMs?
`2017-Jan-Logic-2`
   a. They do not need decoders to select the data because they do not share output wires since the charge on the capacitors is so small
   b. Each bit is slower because it only has a small amount of charge from a capacitor which has to drive the output
   c. Each bit is smaller because it is stored in a capacitor instead of several transistors
   d. All of the above

DRAMs do share output wires just like SRAMs and therefore need the same kinds of decoders. The bits are slower because they have less charge to read, but this doesn't make them cheaper. However, because each bit is smaller (since it goes down into a trench in the silicon instead of using many transistors on the top) manufacturers can fit more bits on each chip, which makes them cheaper.

9. What does the decoder do in a memory?
`2017-Jan-Logic-3`
   a. Take in the row to read and produce the address
   b. Take in the address and choose the row to read
   c. Take in the address and select the columns to output
   d. Take in the columns and produce the address

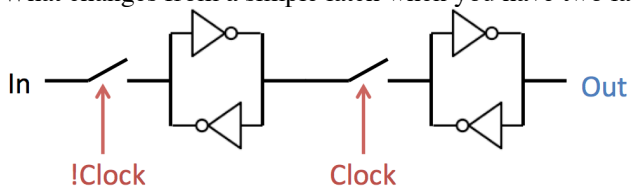
The decoder takes the address and selects the row to read. We then use a mux to choose which of the columns to output.

10. What does a latch do?
`2017-Jan-Logic-4`
   a. Store a value
   b. Update the stored value on the clock signal

c. Break the feedback loop in a sequential circuit
d. All of the above

A latch does all of these things by only transferring its input to the output when the clock changes.

11. What changes from a simple latch when you have two latches back to back with opposite clock signals?



2017-Jan-Logic-5

a. The latch behaves the same but it takes longer for the data to get through
b. The data is only updated when the clock goes from low to high
c. The data only goes through the latch when the clock is low
d. The latch can store a value when the clock is low

The second latch causes the output value to only update when the clock changes from low to high. This prevents the data from going straight through the latch at any time.

12. A designer connects a long chain of AND gates together. (E.g., the output from the first AND goes into one of the inputs of the second AND and so forth.) The circuit uses 1.0V to represent 1 and 0.0V to represent 0. When the circuit comes near a mobile phone it experiences +0.1V of noise on the wires between each AND gate. How many AND gates can be put in a chain until the noise can cause the wrong result?

2017-Jan-Logic-6

a. 0
b. 1
c. 5
d. No limit

At each AND gate the inputs are evaluated as 0 if the are < half way (<0.5) and 1 if the are > (>=0.5V). Therefore, the +0.1V that appears on the output will not affect the next AND gate, and each AND gate will eliminate the previous noise, so you can put together as many AND gates as you want. This is why we can build huge digital systems.

13. What happens if you activate two rows in a memory array at the same time?

2014-Apr-Logic-1

a. You corrupt the data you are reading
b. You can read out twice as much data
c. You can read and write at the same time
d. You can read and write at the same time as long as it is to different rows

Each column output wire is shared with all the rows, so if you activate multiple rows at the same time their output (or input) gets mixed on the wires and will corrupt their values.

14. Why is DRAM so much cheaper than SRAM?

2014-Apr-Logic-2

a. DRAM doesn't require electricity to retain its values
b. DRAM memory cells are much smaller than SRAM cells
c. DRAM doesn't require transistors to work
d. All of the above

DRAM is cheaper because it is smaller.

15. As circuits get smaller, the size of the DRAM cells is shrinking. This means there are fewer and fewer electrons to store each bit in the DRAM. What implications does this trend have?

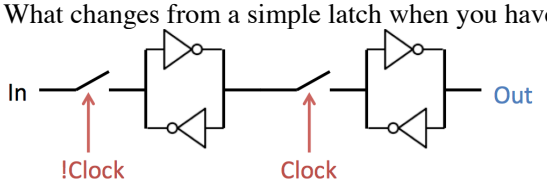
2014-Apr-Logic-3

a. The DRAM will become less reliable since it is harder to know whether you read out a 1 or 0 if there are fewer electrons.
b. The DRAM will become slower since you have to wait longer to get enough of the electrons out to see if it is a 0 or 1.
c. The DRAM will be more sensitive to cosmic rays that leave random electrons on the chip.

d. All of the above

All of the above. It is becoming harder and harder to build DRAMs that work quickly and reliably as a result.

16. What changes from a simple latch when you have two latches back to back with opposite clock signals?



2014-Apr-Logic-4

a. The latch behaves the same but it takes longer for the data to get through
b. The data is only updated when the clock goes from low to high
c. The data only goes through the latch when the clock is low
d. The latch can store a value when the clock is low

The second latch causes the output value to only update when the clock changes from low to high. This prevents the data from going straight through the latch at any time.

17. What is the current state in a counter circuit?

2014-Apr-Logic-5

a. The adder
b. The next count
c. The current count
d. The clock

The state for an adder circuit is the current count. The adder is the combinational logic that generates the next state (the next count) and the clock is the signal that causes the next state to be stored and replace the current state.

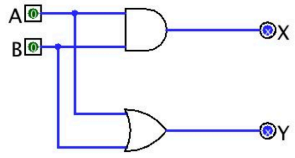
18. What would happen if you built a counter circuit without a state memory (e.g., latch or flip-flop)?

2014-Apr-Logic-6

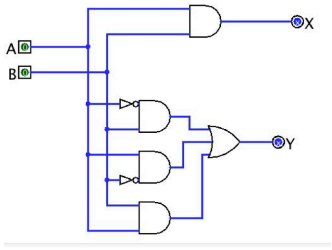
a. It would work without a clock
b. The signals would feedback around and count uncontrollably
c. You couldn't calculate the next state
d. All of the above

The signals would feedback around as fast as the logic would allow them so it would uncontrollably update the count. It wouldn't "work", but it would constantly calculate the next state.

19. These two circuits do the same thing. If the inputs to all gates change at the same time, which of them will be faster?



Circuit A



Circuit B

2015-Oct-Logic-1

a. X output: Circuit A, Y output: Circuit A
b. X output: Circuit B, Y output: Circuit A
c. X output: Both the same, Y output: Circuit A
d. X output: Both the same, Y output: Circuit B

The circuits do the same thing for the X output so they will be the same speed (on AND gate). For the Y output, circuit B's longest path is an inverter->AND->OR, while circuit A just has one OR gate. As a result circuit A has the shorter path and will be faster.

20. Why are DRAMs slower than SRAMs?

2015-Oct-Logic-2

a. They only have small transistors to generate the output

b. They only have a small amount of charge to output
c. They have longer wires for the output
d. They aren't

DRAMs store their values in small capacitors, which means they only have a very small amount of charge to use to output their value. SRAMs have small transistors to generate the output, but those can still push out a lot more charge than DRAMs because they are attached to the power supply. In both cases the effect of the wire length is the same.

21. What does the decoder do in a memory?

`2015-Oct-Logic-3`

a. Take in the row to read and produce the address
b. Take in the address and choose the row to read
c. Take in the address and select the columns to output
d. Take in the columns and produce the address

The decoder takes the address and selects the row to read. We then use a mux to choose which of the columns to output.

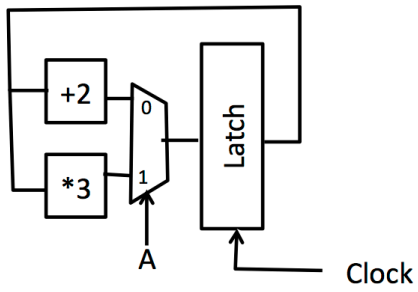
22. Why are binary circuits more immune to noise than analog circuit?

`2015-Oct-Logic-4`

a. Binary values are further apart in terms of voltage than analog values, so they are harder to confuse
b. Binary circuits generate un-polluted outputs (e.g., 1 or 0 out) after each circuit, so the noise doesn't propagate through
c. Binary circuits interpret values close to a 1 or 0 as a 1 or 0 even if they are not exactly 1 or 0
d. All of the above

All of the above.

23. In the circuit below, an adder takes 10ps, a multiplier 50ps, a latch 10ps, and a mux 1ps.
What will happen if you run the clock at 25ps for the above circuit?

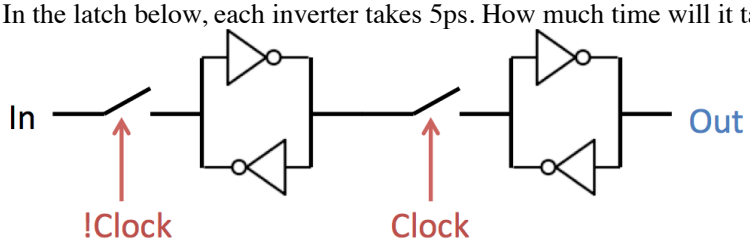


`2015-Oct-Logic-5`

a. It will always produce the wrong results
b. It will always produce the right results
c. It will only produce the right results if A=0
d. It will only produce the right results if A=1

The A=0 path (adder->MUX->latch) needs 21ps, and the clock is 25ps so that is not too fast. The A=1 path (multiplier->MUX->latch) needs 61ps, which is longer than the 25ps clock, so it will not produce the right value in time.

24. In the latch below, each inverter takes 5ps. How much time will it take after the clock rises before the output changes?



`2015-Oct-Logic-6`

a. 5ps
b. 10ps
c. 15ps
d. 20ps

The output will change after going through just one inverter. The second output inverter is only needed when the clock goes low again to keep the value there.

25. What output will you get if you set the inputs to a 4-bit encoder to 1100? (that is, inputs 3 and 4 are set to 1)
`2018-Oct-Logic-1`
    a. Output will be 12 in binary
    b. Output will be 4 in binary
    c. The circuit will generate a valid binary output, but we can't tell what it will be
    d. The circuit will generate a non-binary output

An encoder expects a one-hot input. This is not one-hot, so we don't know what the internal logic in the circuit will do. However, we do know that the 2 bits of output will have values of 1 or 0 each, so the result will be a valid binary number.

26. How many inputs and outputs will an 8-bit DEMUX with a 4-bit select signal have?
`2018-Oct-Logic-2`
    a. 16 inputs and 1 output, each 4-bits wide
    b. 1 input and 16 outputs, each 4-bits wide
    c. 1 input and 16 outputs, each 8-bits wide
    d. 16 inputs and 1 output, each 8-bits wide

A DEMUX takes one input and sends it to one of many outputs. The width tells you how wide the input and output signals are. The 4-bit select signal tells you there will be 2^4=16 possible outputs.

27. What does the decoder do in a memory?
`2016-Aug-Logic-1`
    a. Take in the row to read and produce the address
    b. Take in the address and choose the row to read
    c. Take in the address and select the columns to output
    d. Take in the columns and produce the address

The decoder takes the address and selects the row to read. We then use a mux to choose which of the columns to output.

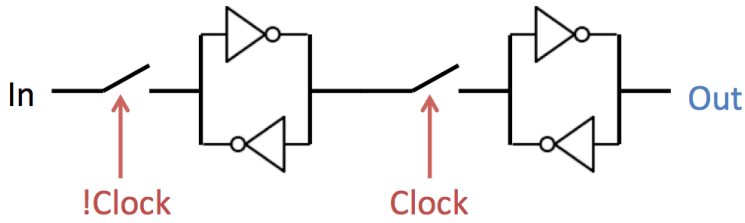
28. Why are DRAMs so much cheaper than SRAMs?
`2016-Aug-Logic-2`
    a. They do not need decoders to select the data because they do not share output wires since the charge on the capacitors is so small
    b. Each bit is slower because it only has a small amount of charge from a capacitor which has to drive the output
    c. Each bit is smaller because it is stored in a capacitor instead of several transistors
    d. All of the above

DRAMs do share output wires just like SRAMs and therefore need the same kinds of decoders. The bits are slower because they have less charge to read, but this doesn't make them cheaper. However, because each bit is smaller (since it goes down into a trench in the silicon instead of using many transistors on the top) manufacturers can fit more bits on each chip, which makes them cheaper.

29. How many inputs will a 4-bit mux with a 3-bit select signal have?
`2016-Aug-Logic-3`
    a. 3 inputs each 4 bits wide
    b. 4 inputs each 3 bits wide
    c. 8 inputs each 4 bits wide
    d. 16 inputs each 3 bits wide

The 3 bit select signal will select between 23=8 possible inputs. So there are 8 inputs, each of which is 4 bits wide.

30. What changes from a simple latch when you have two latches back to back with opposite clock signals?



2016-Aug-Logic-4

a. The latch behaves the same but it takes longer for the data to get through
b. The data is only updated when the clock goes from low to high
c. The data only goes through the latch when the clock is low
d. The latch can store a value when the clock is low

The second latch causes the output value to only update when the clock changes from low to high. This prevents the data from going straight through the latch at any time.

31. Is it easier to convert from an optimized circuit to a truth table or from a truth table to an optimized circuit?

2016-Aug-Logic-5

a. Easier to go from an optimized circuit to truth table
b. Easier to go from a truth table to an optimized circuit
c. Both the same
d. Trick question: if you can use a Karnaugh map then it is the same, otherwise it is harder to convert to a circuit

It is generally harder to convert form a truth table to an optimized circuit because there are so many possible circuits for each truth table. Karnaugh maps are one way of helping, but they are not less complicated than brute-forcing a truth table.

32. What does a latch do?

2016-Aug-Logic-6

a. Store a value
b. Update the stored value on the clock signal
c. Break the feedback loop in a sequential circuit
d. All of the above

A latch does all of these things by only transferring its input to the output when the clock changes.

33. What will be the value of R12 after executing the following two instructions?

```
and R12, R12, R12
or R12, R12, R12
```

2014-Dec-Logic-1

a. 0000 (Zero)
b. FFFF (All ones)
c. It depends on the value of R12 before the first instruction
d. The value of R12 will not change.

The logic operations AND and OR are idempotent, this means that A AND A = A, A OR A = A, therefore, the value will not change.

34. How many outputs will a binary adder that adds two 4-bit values have?

2014-Dec-Logic-2

a. 4
b. 5
c. 8
d. 9

4 bits of output plus 1 carry out.

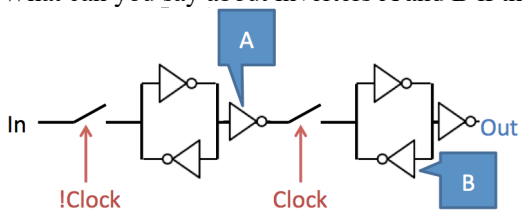
35. What is the output of a 4-bit encoder with the input 0011?

2014-Dec-Logic-3

a. 3
b. Binary 11
c. Output 3 is 1, all other outputs are 0
d. Undefined

An encoder takes in a 1-hot signal and outputs the binary value. If the input is not 1-hot (0011 is not 1-hot as two values are set) then the output is undefined.

36. What can you say about inverters A and B if this circuit is to work as an edge-triggered latch?

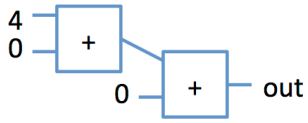


2014-Dec-Logic-4

a. They need to be the same
b. A needs to be smaller (weaker) than B
c. A needs to be bigger (stronger) than B
d. Depends on the value being stored and the value that was there before

For the value from the first latch to be written into the second latch, the output has to be strong enough to overwrite the value that is in the latch before it. This means that inverter A has to be bigger (faster) than inverter B so it can overwrite it.

37. A designer builds a 3-bit digital adder that works pretty well, but has +10%V of noise on each output. The designer connects 4 and 0 to one adder, and then uses a second adder to add that result to 0. What are the result and the decimal interpretation?



2014-Dec-Logic-5

a. Result=1V 0V 0V. Interpretation=4
b. Result=1.1V 0V 0V. Interpretation=4
c. Result=1.11V 0V 0V. Interpretation=4
d. Result=4.84V 0V 0V. Interpretation=5

The first adder outputs 1.1V 0V 0V because the 1V output for the 3rd bit (100=4) will be +10%. The second adder interprets this as 100 and then outputs 100+000=100, but also adds 10% to the output 1 for 1.1V 0V 0V. In both cases the result is interpreted as 100 or 4.

38. Cosmic rays (alpha particles) carry charge and pass through the earth regularly. What kind of memory is more likely to be corrupted by an alpha particle hitting it (a strike), and why?

2014-Dec-Logic-6

a. SRAM: the inverters are very small so they aren't powerful enough to overcome it
b. DRAM: the capacitor is very deep so it is a bigger target for a particle coming from space
c. SRAM: the feedback inverters mean that any small change will be amplified
d. DRAM: the capacitor holds very few electrons so the particle doesn't have change the charge much to change the value

The biggest issue is the amount of charge that has to be changed to flip the bit. In DRAM, the capacitors don't hold a lot of charge (they are small to fit in a lot of bits) so it is easier to switch their value. The SRAM cell automatically regenerates the value from both sides (feedback) so a hit has to be strong enough to overcome one of the inverters.

39. How many output bits does an adder have if it is adding two 8-bit numbers?

2015-Apr-Logic-1

a. 5
b. 8
c. 9
d. Depends on whether the numbers are two's compliment or fixed-point

8 bits plus 1 carry out, for 9 bits. This is the same for two's compliment.

40. (!A AND B) OR (!B AND A) is the equivalent of which of the following?

2015-Apr-Logic-2

a. A AND B?

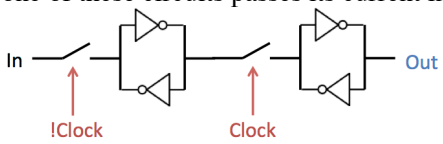
b. A OR B
c. A XOR B?
d. A XNOR B

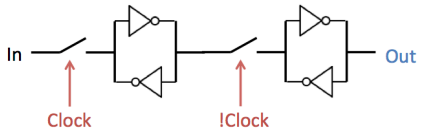
The logic equation says that A and B are different, which is XOR.

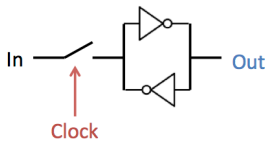
41. Why are transparent latches worse than edge-triggered flipflops?
2015-Apr-Logic-3
    a. They are slower. They look if the clock signal is high or low the whole time, instead of just looking at the edge of the clock
    b. They don't avoid feedback loops as they allow the input to pass directly through to the output
    c. They don't allow the input to pass directly through to the output, so they slow down the circuit
    d. They are bigger because they need more than two inverters

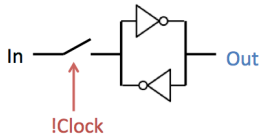
Edge-triggered flipflops use two latches back-to-back to load the input into the first latch when the clock is low and then transfer it to the second latch when the clock is high. This prevents the signal from feeding straight through and avoids feedback through the circuit.

42. Which one of these circuits passes its current input to the output whenever the clock goes high?

a.


b.


c.


d.


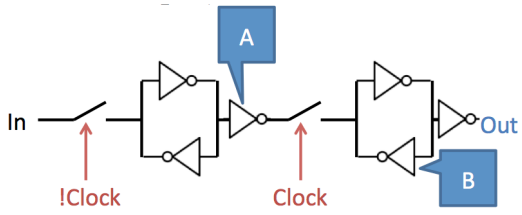
2015-Apr-Logic-4
    a. !Clock/Clock
    b. Clock/!Clock
    c. Clock
    d. !Clock

The single latch with clock as the input control will get new data in whenever the clock goes high, and will constantly output that result.

43. Why is DRAM slower than SRAM?
2015-Apr-Logic-5
    a. There is a lot more of it, so it takes longer to get data from it to the processor
    b. It's built of capacitors which have less charge and take longer to produce output
    c. It's built of feedback loops which need external power to output their data
    d. The wires are longer so it takes longer to charge up the wire and read the output

DRAM stores charge in capacitors and they are tiny, so there is only a tiny amount of charge. Detecting the change in output from this tiny amount of charge is slow. SRAM uses feedback loops, which do require power, but this means they can drive their output much faster. The wire lengths are not dramatically different.

44. How would the output of this circuit change if we removed the inverters at the end of each feedback loop? (One is labeled A, the other is directly before the Out. Ignore the B label.)



2015-Apr-Logic-6

a. The output would be inverted, but faster
b. The output would be inverted, but slower
c. <u>The output would not change, but be faster</u>
d. The output would not change, but be slower

There are two inverters in a row, so they cancel each other out. Removing them will not change the output, but it will make the circuit faster since there are fewer gates for the signal to go through.

45. A XOR B is the equivalent of which of the following?

2013-Jan-Logic-1

a. <u>(A AND !B) OR (!A AND B)</u>
b. (!A AND !B) OR (A AND B)
c. (A AND !B) AND (!A AND B)
d. (!A AND !B) AND (A AND B)

46. How many control bits are needed to control a Multiplexor with 8 inputs?

2013-Jan-Logic-2

a. 2
b. <u>3</u>
c. 8
d. 2^8

47. The combinational logic does what in a sequential circuit?

2013-Jan-Logic-3

a. Store the current value
b. Store the next value
c. Calculate the current value
d. <u>Calculate the next value</u>

48. What limits the speed of a sequential circuit?

2013-Jan-Logic-4

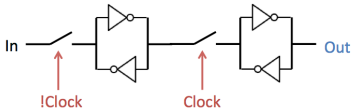
a. The slowest latch
b. The slowest path through the combinational logic
c. <u>The slowest latch + the slowest path through the combinational logic</u>
d. The longest instruction

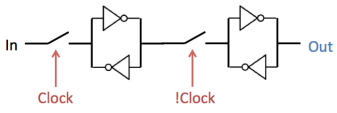
49. Which is faster, and why? DRAM or SRAM?

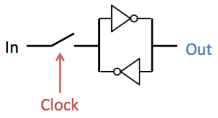
2013-Jan-Logic-5

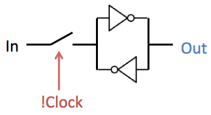
a. DRAM: each bit is stored in a powered feedback loop so it can be read quickly
b. DRAM: each bit is stored in a passive capacitor so it can be read quickly
c. <u>SRAM: each bit is stored in a powered feedback loop so it can be read quickly</u>
d. SRAM: each bit is stored in a passive capacitor so it can be read quickly

50. Which one of these circuits updates its output only when the clock goes from 1 to 0?

a. 

b. 

c. 

d. 

`2013-Jan-Logic-6`

a. Clock
b. !Clock
c. !Clock / Clock
d. Clock / !Clock

51. What does the clock do in a counter circuit?

`2015-Aug-Logic-1`

a. Control when the computed next value is stored as the current value
b. Control the storage element to prevent a feedback loop
c. Make sure the next value logic has enough time to compute the next value
d. All of the above

The clock is responsible for telling the storage element to take in the next value and store it as the current value. The transition of the clock is what allows the data to go through the storage element without forming a feedback loop. The clock has to be slow enough to give the logic enough time to compute the next value.

52. Which is faster: DRAM or SRAM and why?

`2015-Aug-Logic-2`

a. DRAM: it uses capacitors to drive the signal onto the output wire
b. DRAM: it uses a powered feedback loop to drive the signal onto the output wire
c. SRAM: it uses capacitors to drive the signal onto the output wire
d. SRAM: it uses a powered feedback loop to send the signal onto the output wire

SRAM uses a powered feedback loop to store the bit, which means it can drive the output much faster (it can use electrons from the power supply), while DRAM stores the bit as charge in a capacitor and can only use whatever charge is stored there to drive the output.

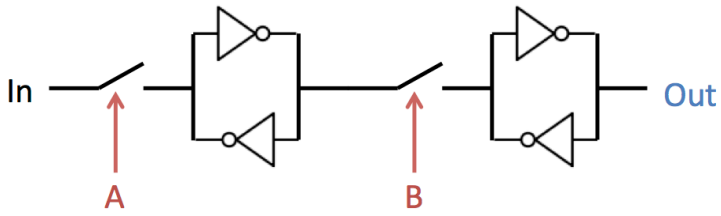
53. What will be the value of R12 after executing the following two instructions?
and R12, R12, R12
or R12, R12, R0

`2015-Aug-Logic-3`

a. 0x0000 (Zero)
b. 0xFFFF (All ones)
c. It depends on the value of R12 before the first instruction
d. The value of R12 will not change

The logic operations AND and OR are idempotent, this means that A AND A = A, A OR A = A. But A OR 0 never changes anything.

54. What signals do you put into A and B to build a latch that stores data on the rising edge of the clock?

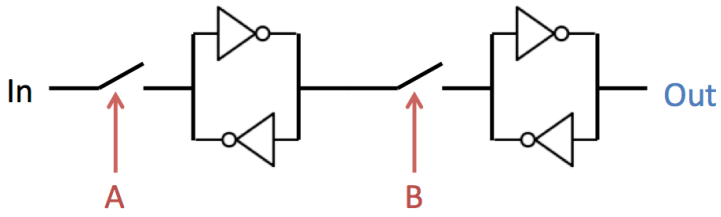


2015-Aug-Logic-4

a. A=CLK, B=!CLK
b. A=CLK, B=CLK
c. A=!CLK, B=CLK
d. A=!CLK, B=!CLK

When clock is low you want the data to go into the first latch, and then when clock goes high it should transition to the second. This means A should be closed when the clock is low (A=!CLK) and B when the clock is high (B=CLK).

55. Which is true about the circuit below if both A and B are attached to CLK?



2015-Aug-Logic-5

a. The circuit can cause feedback when the CLK is low
b. The circuit is slower
c. A and B
d. Neither

When the clock is low the gates are opened so no data can pass through. (You do get data passing through when the clock is high.) The circuit is slower because the data has to go through two sets of latches.

56. A designer connects a long chain of AND gates together. (E.g., the output from the first AND goes into one of the inputs of the second AND and so forth.) The circuit uses 1.0V to represent 1 and 0.0V to represent 0. When the circuit comes near a mobile phone it experiences +0.1V of noise on the wires between each AND gate. How many AND gates can be put in a chain until the noise can cause the wrong result?

2015-Aug-Logic-6

a. 0
b. 1
c. 5
d. No limit

At each AND gate the inputs are evaluated as 0 if the are < half way (<0.5) and 1 if the are > (>=0.5V). Therefore, the +0.1V that appears on the output will not affect the next AND gate, and each AND gate will eliminate the previous noise, so you can put together as many AND gates as you want. This is why we can build huge digital systems.

57. What is true about a logic circuit derived using a Karnaugh map vs. one derived from a truth table?

2014-Sep-Logic-1

a. Karnaugh maps always produce simpler circuits
b. Karnaugh maps usually produce simpler circuits
c. Karnaugh maps always produce the same circuit
d. Truth tables always produce simpler circuits

Karnaugh maps usually produce simpler circuits because they allow you group outputs that are "closer" together in terms of their inputs. This is because the inputs are re-arranged to have fewer changes between each output row/column. Sometimes the Karnaugh map produces the same circuit as the truth table, and they always produce the same logic function, even if the circuits are different.

58. Which of the following instructions will store 0 in R3?

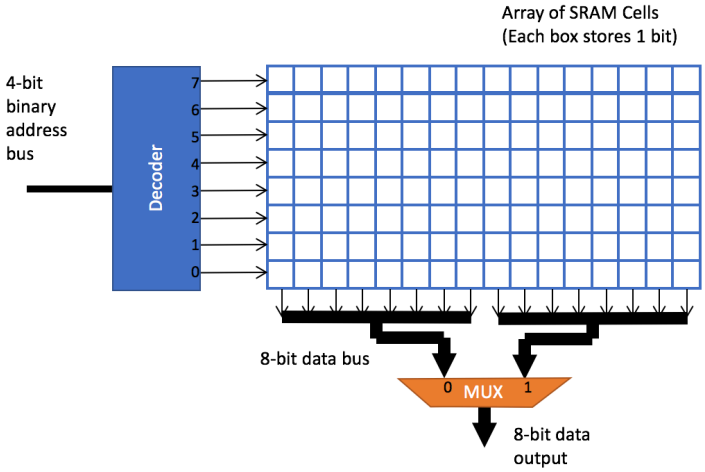
2014-Sep-Logic-2

a. and R3, R3, R3

b. or R3, R3, R3
c. xor R3, R3, R3
d. None of the above

XOR is true when two bits are different. Since all the bits in R3 are the same as the bits in R3, none of them will be different and the result will be all 0s.

59. What would happen if you replaced the decoder in the SRAM design with an encoder?

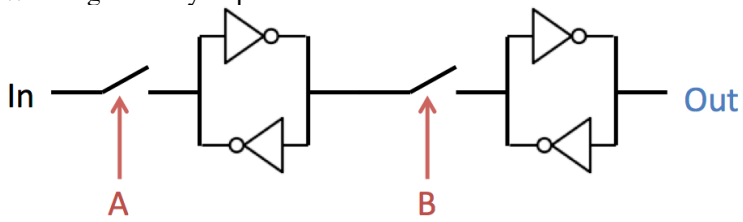


2014-Sep-Logic-3

a. It would activate different rows, but it would still work
b. It would activate different rows, and would not work
c. It would activate multiple rows, but would still work
d. It would activate multiple rows, and would not work

An encoder could have multiple outputs active at the same time. E.g., if input #3 was selected then outputs 0 and 1 would both be true. This would cause the SRAM array to output data from rows 0 and 1 at the same time, and since they share output wires, their values would corrupt each other. The decoder was used because it has a one-hot output.

60. What signals do you put into A and B to build a latch that stores data on the falling edge of the clock?

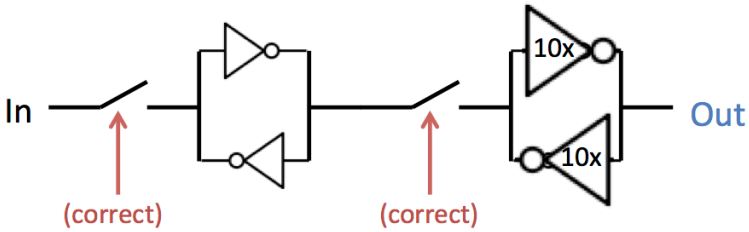


2014-Sep-Logic-4

a. A=CLK, B=!CLK
b. A=CLK, B=CLK
c. A=!CLK, B=CLK
d. A=!CLK, B=!CLK

When clock is high you want the data to go into the first latch, and then when clock goes low it should transition to the second. This means A should be closed when the clock is high (A=CLK) and B when the clock is low (B=!CLK).

61. A designer makes a mistake on a positive-edge triggered flip-flop and makes the inverters in the second latch 10x bigger (10x more powerful, and about 3x faster) than the first latch. What happens? (Assume the clocks are connected correctly.)



2014-Sep-Logic-5

a. The latch works the same as before, but it is a faster due to the more powerful inverter
b. The first latch cannot overwrite the value in the second latch so the flip-flop doesn't work
c. The second latch cannot overwrite the value in the first latch so the flip-flop doesn't work
d. None of the above

The second latch is so much larger that when the second gate is closed (on the positive clock signal) the first latch isn't powerful enough to change the value in the second latch, so the data does not get stored in the flip flop.

62. Why are DRAMs so much cheaper than SRAMs?
`2014-Sep-Logic-6`
    a. They do not need decoders to select the data because they do not share output wires since the charge on the capacitors is so small
    b. Each bit is slower because it only has a small amount of charge from a capacitor which has to drive the output
    c. Each bit is smaller because it is stored in a trench deep down into the silicon instead of several transistors spread out on top
    d. All of the above

DRAMs do share output wires just like SRAMs and therefore need the same kinds of decoders. The bits are slower because they have less charge to read, but this doesn't make them cheaper. However, because each bit is smaller (since it goes down into a trench in the silicon instead of using many transistors on the top) manufacturers can fit more bits on each chip, which makes them cheaper.

63. Why is DRAM so much cheaper than SRAM?
`2016-Oct-Logic-1`
    a. DRAM doesn't require electricity to retain its values
    b. DRAM memory cells are much smaller than SRAM cells
    c. DRAM doesn't require transistors to work
    d. All of the above

DRAM is cheaper because it is smaller.

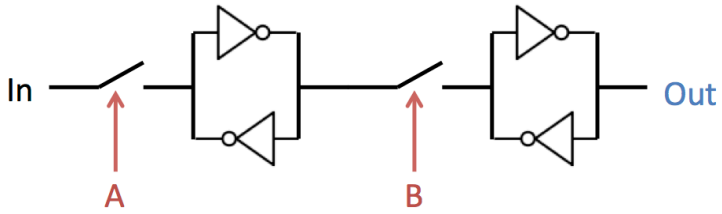
64. Will a MUX function correctly if more than one input is active (valid) at a time, and why?
`2016-Oct-Logic-2`
    a. No, output is undefined or invalid
    b. No, a MUX can only have one input at a time
    c. Maybe, but it will depend on how the MUX is built internally
    d. Yes, output is always the selected input

All the inputs to a MUX are valid; the MUX just selects which one to pass on to the output.

65. A MUX switches between 4 3-bit inputs. How many wires go into the MUX?
`2016-Oct-Logic-3`
    a. 3
    b. 5
    c. 12
    d. 14

The MUX has 4*3 input signal wires, plus two input wires to select from the four inputs, for 14 total.

66. What signals would you put into A and B to build a circuit that allows data to pass through only when the clock is low? (Putting a 1 into the gate closes it.)



`2016-Oct-Logic-4`
    a. A=CLK, B=CLK
    b. A=CLK, B=!CLK
    c. A=!CLK, B=CLK
    d. A=!CLK, B=!CLK

With !CLK into both, the gates will be closed when CLK=0, which means the value will pass all the way through when CLK is low..

67. What limits the speed of a sequential circuit?
`2016-Oct-Logic-5`
    a. The longest instruction
    b. The slowest latch
    c. The slowest path through the combinational logic
    d. The slowest path through the combinational logic and latch

The clock needs to wait for the longest (slowest) combinational path and the time it takes to store the data in the latch at the end.

68. Are all circuits that have the same truth table equivalent in all regards?
`2016-Oct-Logic-6`
    a. Yes, their outputs will be the same by definition
    b. Yes, there is only one way to create a circuit for a given truth table
    c. No, different implementations may be slower or need more gates
    d. No, different implementations will have different outputs

You can implement a given function many different ways, but some of them will be far less efficient than others.

69. How many output bits does an adder have it if is adding two 16-bit two's complement numbers?
`2016-Jan-Logic-1`
    a. 16
    b. 17
    c. 32
    d. 33

An adder for two's complement numbers is the same as one for unsigned. The adder will take in two 16-bit values and output a 16-bit value and a carry out, or 17 bits total.

70. How many control bits are needed to control a 4-bit MUX with 8 inputs?
`2016-Jan-Logic-2`
    a. 2
    b. 3
    c. 4
    d. 8

To select among the 8 inputs you need 2^3 combinations, which requires 3 control bits.

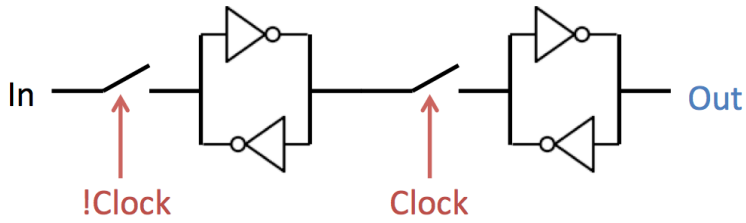
71. Why is SRAM more expensive than DRAM?
`2016-Jan-Logic-3`
    a. SRAM requires electricity to retain its values
    b. SRAM requires transistors to work
    c. SRAM memory cells are larger than DRAM's
    d. All of the above

SRAM is more expensive because it requires more area on the silicon die to store data. This is because the memory cells require 6 transistors each, compared to 1 transistor and a capacitor for DRAM.

72. What is the output of a 4-bit encoder with the input 0000?
`2016-Jan-Logic-4`
    a. 00
    b. Output 0 is 1, all others are 0
    c. All outputs are 1
    d. Undefined

An encoder will take in 0000 and output the binary equivalent of it, which is 0=00.

73. What is the benefit of having two latches back to back with opposite clock signals controlling their inputs?



2016-Jan-Logic-5

a. The data is only updated when the clock goes from low to high
b. The data only goes through the latch when the clock is low
c. The latch can store a value when the clock is high
d. The latch behaves the same as a single latch but is slower

The second latch causes the output value to only update when the clock changes from low to high. This prevents the data from going straight through the latch at any time. Running more slowly is not a benefit, and the regular latch can also store values when the clock is high.

74. As circuits get smaller, the size of the DRAM cells is shrinking. This means there are fewer and fewer electrons to store each bit in the DRAM. What implications does this trend have?

2016-Jan-Logic-6

a. The DRAM will become more reliable since it is easier to know whether you read out a 1 or 0 if there are fewer electrons.
b. The DRAM will become faster since you won't have to wait as long to get all the electrons out to see if it is a 0 or 1.
c. The DRAM will be more sensitive to cosmic rays that leave random electrons on the chip.
d. All of the above

The DRAM will become less reliable since there are fewer electrons and it is harder to know whether you read a 1 or 0. Similarly it will take longer to read as you need to wait longer to get enough electrons. The DRAM will become more sensitive to cosmic rays as there are fewer electrons to hold the value.

75. If you choose two groups of 2 in a Karnaugh map instead of 1 group of 4, what do you get as the result?

2013-Aug-Logic-1

a. The wrong logic equation
b. A logic equation with twice as many terms
c. A logic equation with half as many terms
d. A logic equation with XOR gates

The more groups you choose the more terms you have in the result. If you use two groups of two instead of 1 group of 4, then you will have two terms in the final equation instead of only 1 with the group of 4.

76. How many inputs will a 4-bit mux with a 3-bit select signal have?

2013-Aug-Logic-2

a. 3 inputs each 4 bits wide
b. 4 inputs each 3 bits wide
c. 8 inputs each 4 bits wide
d. 16 inputs each 3 bits wide

The 3 bit select signal will select between 23=8 possible inputs. So there are 8 inputs, each of which is 4 bits wide.

77. Why does a decoder not have input 0 connected to the logic that generates the output?

2013-Aug-Logic-3

a. It does.
b. Input 0 corresponds to binary output 0, which is all false, so we don't use the input to generate any output.
c. You can never have just input 0 active because the inputs are one hot.
d. Zero cannot be represented with a one hot input.

When input zero is true, the output should be 000. So we do not need to use the input to generate all false outputs.

78. Why are edge-triggered flipflops better than transparent latches?

2013-Aug-Logic-4

a. They are faster because they only look at edge of the clock signal instead of the whole time it is high or low
b. They allow the input to pass directly through to the output, so they speed up the circuit

c. They don't allow the input to pass directly through to the output, so they avoid feedback loops
d. They are smaller than transparent latches because they only need two inverters

Edge-triggered flipflops use two latches back-to-back to load the input into the first latch when the clock is low and then transfer it to the second latch when the clock is high. This prevents the signal from feeding straight through and avoids feedback through the circuit.

79. Why is DRAM so much cheaper than SRAM?
`2013-Aug-Logic-5`
a. DRAM cells are made of two inverters so they are smaller
b. DRAM cells are made of a single capacitor so they are smaller
c. DRAM cells are faster so you need fewer of them
d. DRAM cells are larger so they are easier to make

DRAM cells are made of capacitors, which are much smaller than the 6 transistors needed to make a SRAM cell, so you can fit more of them into a chip.

80. Why is SRAM so much faster than DRAM?
`2013-Aug-Logic-6`
a. It isn't.
b. SRAM cells are made of powered transistors so they can output a stronger signal, which means they can send out their data faster
c. SRAM cells are made of capacitors so they are smaller and their output doesn't have to travel as far
d. SRAM uses caches to make it faster

SRAM cells are made of inverters that feedback on each other. These inverters need power to run, which means they can output a much stronger signal than DRAM, where each cell is just a passive capacitor. When they output a stronger signal it takes less time for that signal to reach the end of the wire.

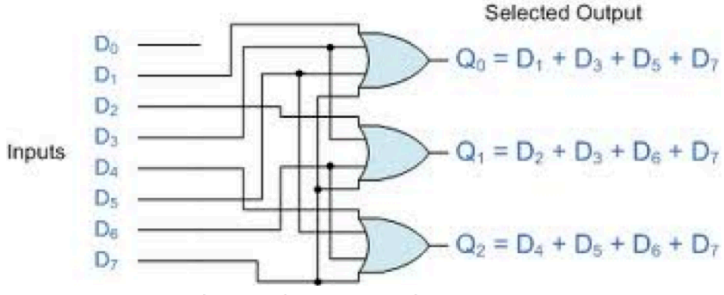
81. How can we use an XOR gate to determine overflow in two's complement math?
`2013-Apr-Logic-1`
a. XOR all output bits together: 1 = overflow
b. XOR carry in to MSB and carry out from MSB: 1 = overflow
c. XOR sign bit of input 1 and sign bit of input 2: 0 = overflow
d. XOR the carry bits together: 1 = overflow

Two's complement math overflows if the carry in to the sign bit (MSB) is different from the carry out.

82. How many inputs would a MUX have that has 3 select bits?
`2013-Apr-Logic-2`
a. 3
b. 6
c. 8
d. 9

3 select bits allow you to select from 2^3 possible inputs, or 8.

83. Why is the D0 input to an encoder not connected to the logic that determines the output?



`2013-Apr-Logic-3`
a. D0 is a dummy input
b. If D0 is true, then we want the outputs to be all false
c. D0 is an illegal input
d. The logic doesn't need D0 as an input because one of the outputs is always true

If the D0 input is 1, then the encoded output should be 000, so the encoder assigns the output to 000 if none of the other inputs are 1, so it can ignore D0.

84. Why does a master-slave latch need two latches?
`2013-Apr-Logic-4`
    a. To prevent the output from going through to the input
    b. To only capture the input on the edge of the clock
    c. To only make the output visible on the edge of the clock
    d. To use feedback to keep the value

The master-slave latch moves the input from the first latch to the second one when the clock changes. This means it captures it on the edge of the clock signal.

85. What kinds of circuits are needed for a counter?
`2013-Apr-Logic-5`
    a. Combinational
    b. State register
    c. Full adder
    d. All of the above

Adders need to store the current count and update it (combinational full adder).

86. What determines the clock speed of a circuit?
`2013-Apr-Logic-6`
    a. The slowest logic gate
    b. The slowest path through the combinational logic
    c. The slowest path through the combinational logic and the register
    d. The number of logic gates in the next stage logic

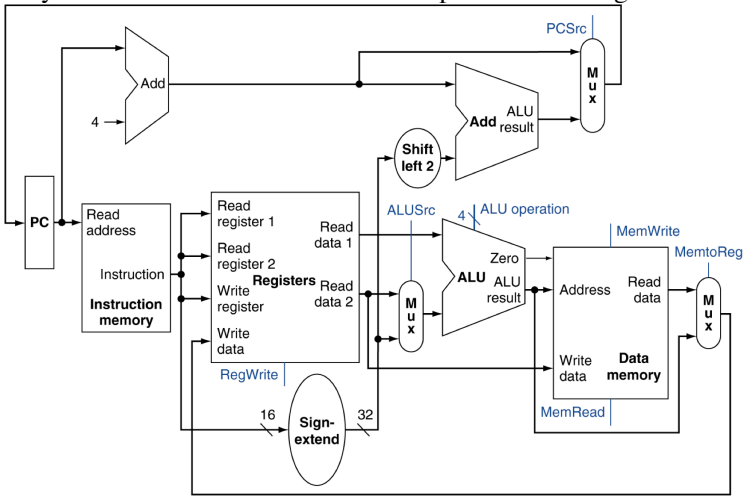
The speed is set by the slowest path through the combinational logic and the register, because if you run any faster not all the signals will be latched in on time.

5. Processor Control and Datapath

1. Which part of the processor does not store state?
`2013-Dec-Processor Control and Datapath-1`
    a. PC
    b. Instruction Memory
    c. Registers
    d. ALU

The ALU is just combinational logic. It takes whatever its inputs are and generates outputs. The other components all store values and update them on the clock signal.

2. Why do we have two adders in the datapath after the register file in the processor below?

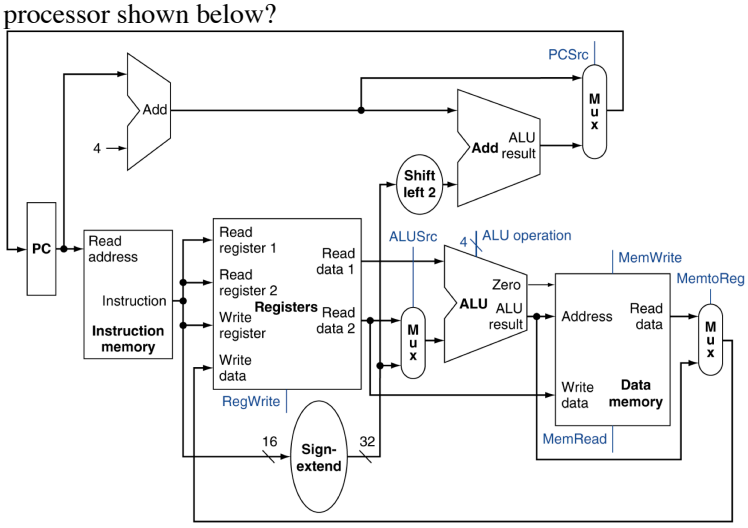


`2013-Dec-Processor Control and Datapath-2`
    a. Need to compute the branch address and whether we should branch at the same time

b. The branch address can't go into the regular ALU
c. Both of the above
d. Neither of the above

We need to both compute the branch address and the branch condition (ALU zero out) at the same time. The branch address (immediate) does indeed go into the regular ALU as it would in an I-type instruction.

3. What does the branch adder (the one with "shift left 2" going into it) compute on an R-type AND instruction in the processor shown below?

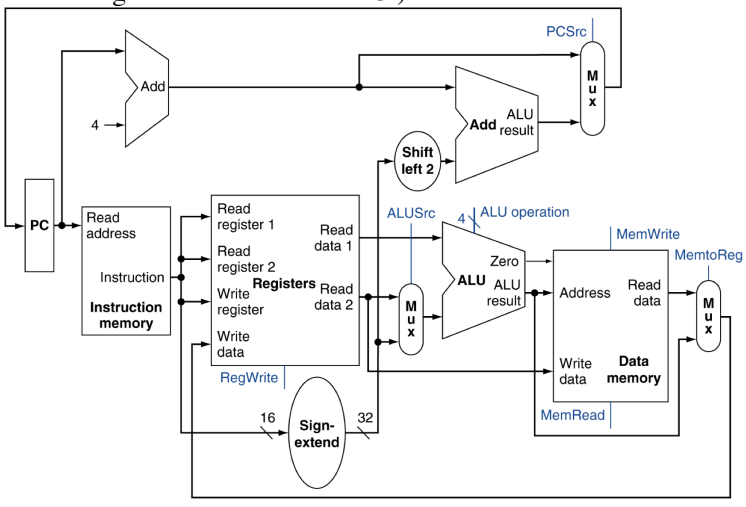


2013-Dec-Processor Control and Datapath-3

a. Nothing
b. The AND of two registers
c. The addition of a the next PC and an immediate
d. The addition of a register and an immediate

The branch adder always adds the (shifted) sign-extended immediate field to the next PC. However, this result is only used if the conditional branch is taken.

4. What are RegWrite and MemtoReg for a sw instruction? (Assume MemtoReg=1 comes from the Data memory and MemtoReg=0 comes rom the ALU.)

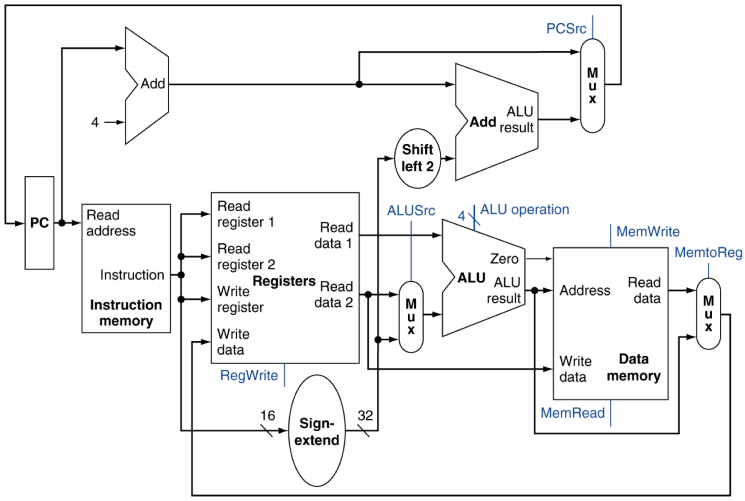


2013-Dec-Processor Control and Datapath-4

a. RegWrite = 1, MemtoReg = X
b. RegWrite = 1, MemtoReg = 0 (can't read while we write, so we have to choose 0)
c. RegWrite = 0, MemtoReg = X
d. RegWrite = 0, MemtoReg = 0 (can't read while we write, so we have to choose 0)

Store word does not write to the register file so we need RegWrite =0. MemtoReg can be anything (X) because we won't use the data we get back. It can even be 1, since we are not setting MemRead=1 to do a read.

5. In the processor below, the ALU takes 20ns, the Instruction Memory 20ns, the Register File 5ns (to read or write), the sign extender 10ns, the Data memory 50ns, the branch adder 5ns, and each MUX 5ns. How long does the slowest instruction take?

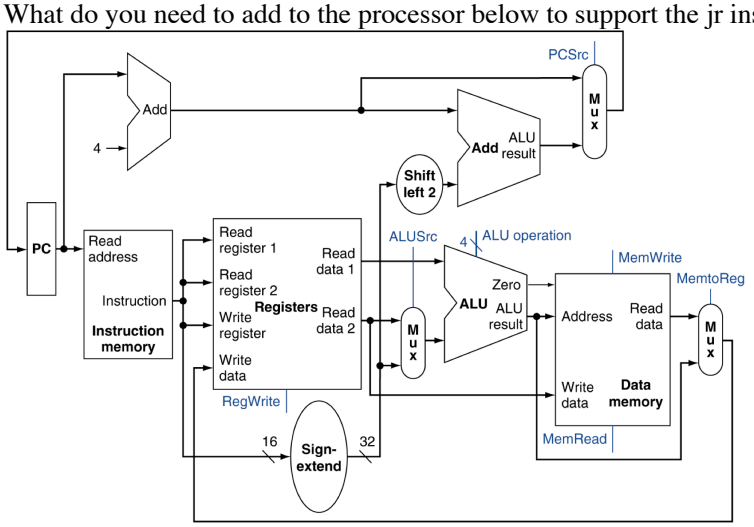


2013-Dec-Processor Control and Datapath-5

a. 105ns
b. 115ns
c. 120ns
d. 125ns

lw will go Instruction Memory, Sign-extension (in parallel with the register file read, but that's shorter so you don't have to add it in), Mux, ALU, data memory, mux, register file write: 20+10+5+20+50+5+5=115.

6. What do you need to add to the processor below to support the jr instruction?



2013-Dec-Processor Control and Datapath-6

a. A path to put 26 bits of the instruction into the next PC
b. A path to put PC+4 into the register file
c. A path to put the output of the register file into the next PC
d. All of the above

For jr we just need to add a path to put the output of the register file into the next PC. (The others are needed for j and jal.)

7. If we were to replace the sign?extend module with a "zero?extend", how would this affect I-type instructions?

2017-Jan-Processor Control and Datapath-1

a. Not at all, the sign?extend is not used for I?type instructions
b. Not at all, the operand is already two's complement encoded
c. If the operand value is negative, then it changes value
d. If the operand value is positive, then it will become negative

Adding 0 to msb of a negative turns it into a different, positive value. Positive values are not affected.

8. A single?cycle processor has three types of instructions: A takes 1ns, B takes 2ns, and C takes 10ns. A program with an instruction mix of 25% A, 70% B, and 5% C will run at what clock speed?

2017-Jan-Processor Control and Datapath-2

a. 1000 MHz (1 / 1ns)

b. 465 MHz (1/2.15ns)
c. 333 MHz (1/3ns)
d. 100 MHz (1/10ns)

The slowest instruction path dictates the clock speed.

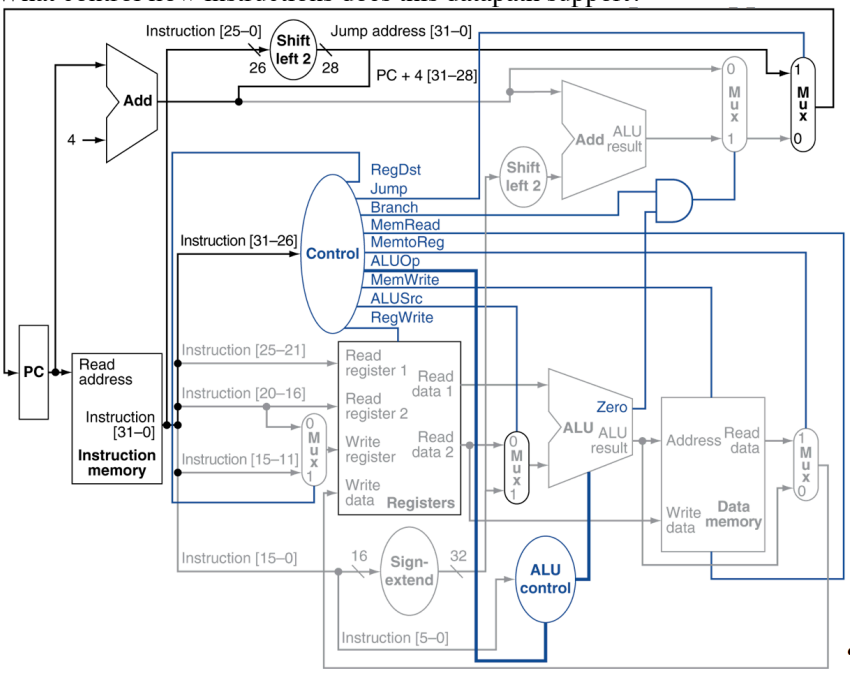
9. Why do we not have hazards in the single-cycle processor?
`2017-Jan-Processor Control and Datapath-3`
   a. Every instruction finishes before the next one starts so the register file always has the right data
   b. No instructions are executing in parallel so there are no conflicts between instructions
   c. Each instruction has access to the whole processor while it is executing so there is no fighting for resources
   d. All of the above

The pipelined register has problems because these things are not true!

10. JAL is a J format instruction. How does the datapath know which register to store the return address into?
`2017-Jan-Processor Control and Datapath-4`
   a. The instruction specifies the destination register
   b. The immediate field specifies the destination for storing the return address
   c. The return address is automatically put in the PC register
   d. The value is generated by the control logic based on the opcode

J format instructions do not have space for anything other than a large constant and the opcode. The destination register needs to be generated by the control logic when it sees that the instruction opcode is JAL.

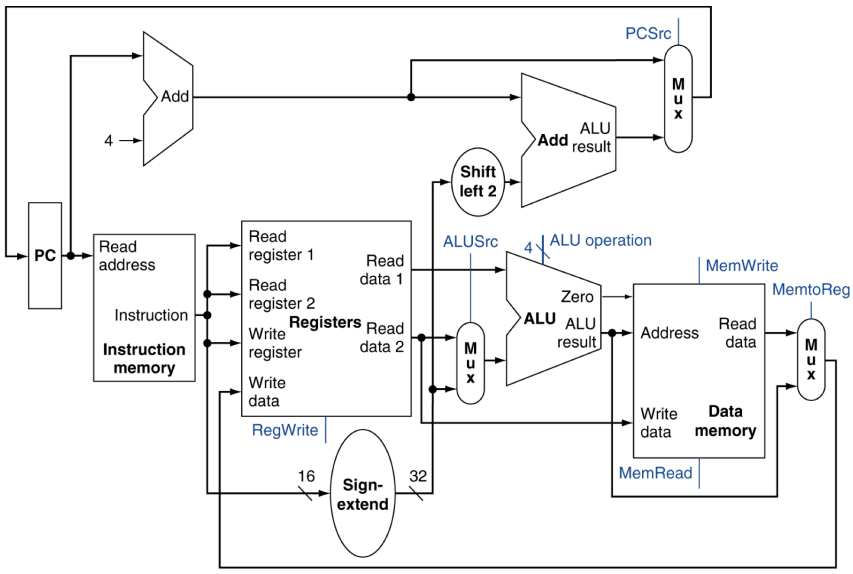
11. What control flow instructions does this datapath support?



`2017-Jan-Processor Control and Datapath-5`
   a. a. BEQ, BNE
   b. b. J, BNE
   c. c. J, BEQ
   d. d. J, BEQ, BNE

The AND gate will only allow us to take the PC+Immediate branch if ALU Zero is true, which means the two values are the same. That is, it only supports BEQ. We support Jump through the top path.

12. What are the control signals in the datapath below for a lw instruction?br>

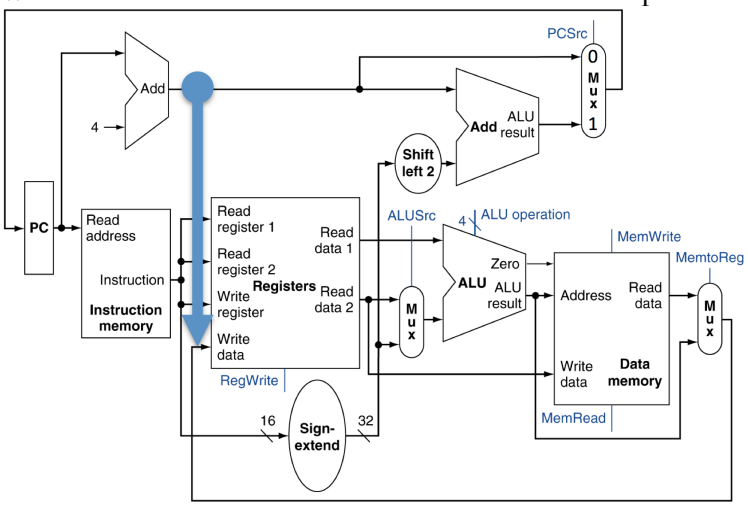


2017-Jan-Processor Control and Datapath-6

a. ALUSrc=SignExt, ALUOp=Add, RegWrite=1, MemRead=1, PCSrc=PC+4
b. ALUSrc=SignExt, ALUOp=Add, RegWrite=1, MemRead=1, PCSrc=ALU
c. ALUSrc=SignExt, ALUOp=NOP, RegWrite =1, MemRead=1, PCSrc=PC+4
d. ALUSrc=Reg, ALUOp=Add, RegWrite =1, MemRead=1, PCSrc=PC+4

The ALU needs to take a register and the sign-extended constant to generate the address (ALUSrc=SignExt, ALUOp=Add) and we need to write the results back to the register file (RegWrite=1) and tell the memory to read (MemRead=1). The next instruction is just PC+4, so PCSrc is PC+4.

13. Which instructions need the connection shown in the datapath below with the thick line?



2014-Apr-Processor Control and Datapath-1

a. jr
b. jal
c. jr and jal
d. jr, jal, and j

Only jal needs this path to write the next PC value into the register file so that jr can return to that address.

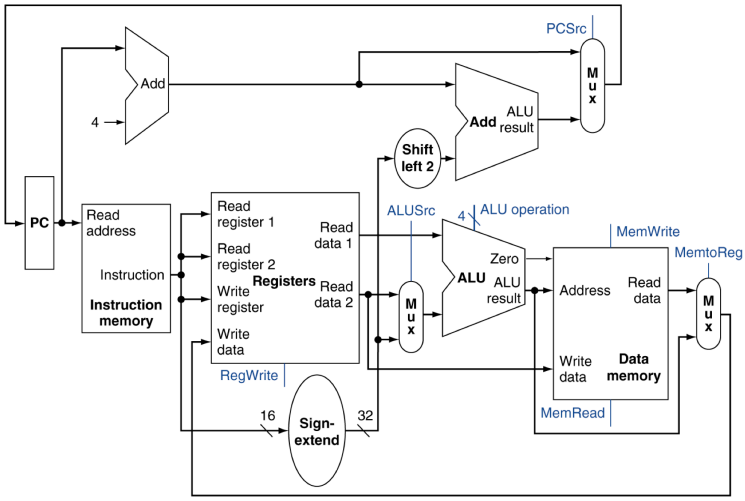
14. Which of the following do not need a clock signal in the datapath?

2014-Apr-Processor Control and Datapath-2

a. Adders
b. MUXes
c. Sign-extender
d. All of the above

All of the above are combinational elements. Only the memories and the registers (including the PC) are sequential elements that need a clock.

15. What are the control signals for bne?

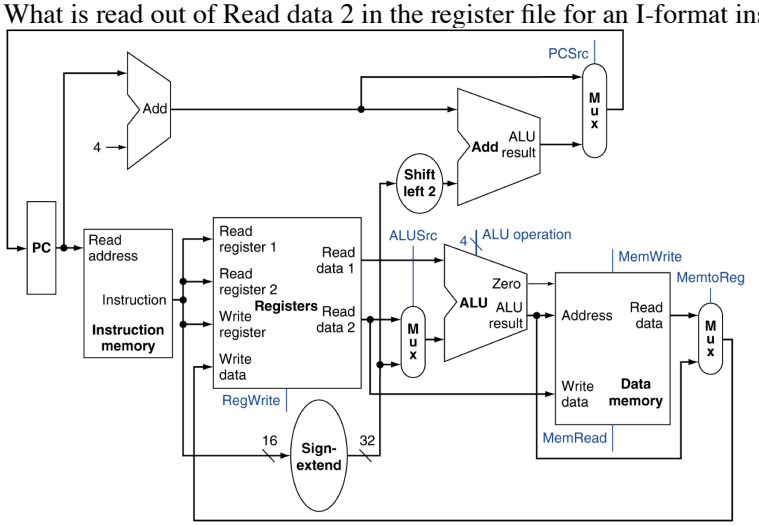


2014-Apr-Processor Control and Datapath-3

a. RegWrite=0, ALUSrc=SignExtend, ALUOp=Sub, PCSrc=!Zero
b. RegWrite=0, ALUSrc= Read data 2, ALUOp=Sub, PCSrc=Zero
c. RegWrite=0, ALUSrc=Read data 2, ALUOp=Sub, PCSrc=!Zero
d. RegWrite=0, ALUSrc=SignExtend, ALUOp=Sub, PCSrc=1

RegWrite=0 since we don't write to the register file. ALUSrc=Read data 2 since we are comparing two registers. ALUOp=Sub to see if the values are different. PCSrc=!Zero, since we want to choose the ALUResult (branch) when they are different, so Zero=0, !Zero=1.

16. What is read out of Read data 2 in the register file for an I-format instruction?



2014-Apr-Processor Control and Datapath-4

a. Nothing
b. The immediate
c. A register based on the immediate value
d. The same register as Read data 1

The Read register 2 input is still 5 bits from the instruction, but now those bits happen to be from the immediate field. This means that Read data 2 outputs some random register based on the immediate value.

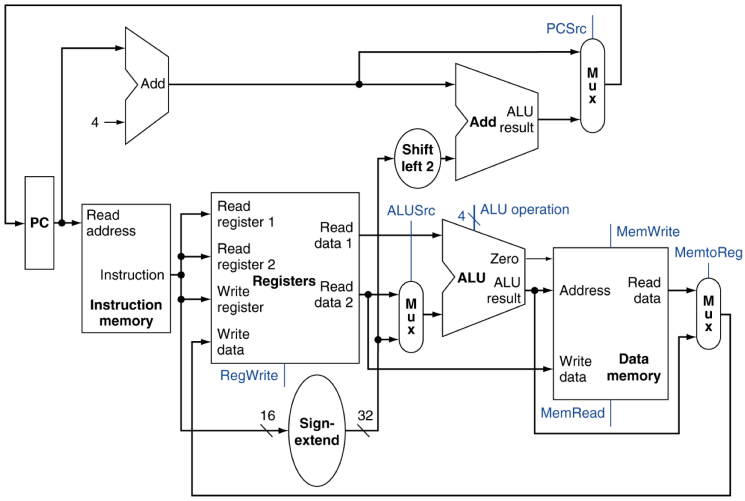
17. Why was a single "branch" output from the control logic not enough to handle both bne and beq instructions?

2014-Apr-Processor Control and Datapath-5

a. We need to know which one looks at the ALU's Zero output
b. We need to interpret the ALU's Zero output differently for each
c. We do different calculations in the ALU for each
d. All of the above

bne and beq interpret the ALU's Zero output differently. A Zero output true means the values are the same, so it is a branch for a beq but not for a bne. In both cases the ALU operation is the same.

18. Why is the path that determines the clock speed of a processor typically determined by the lw or sw instructions?



2014-Apr-Processor Control and Datapath-6

a. They need to use the sign extension logic
b. They use the ALU
c. They use the memory
d. They only read one register file output

They are slow because they have to access the memory and the memory is typically far slower than the rest of the processor. (Hence the reason for caches.) Many other instructions use the sign extension and the ALU, and reading just one register output is not any slower than reading both.

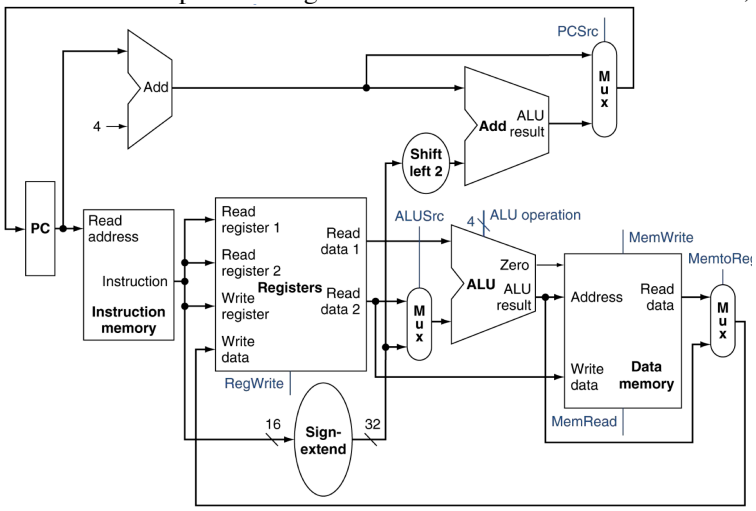
19. How many inputs can a 6-bit multiplexer with 4 control bits switch between?

2015-Oct-Processor Control and Datapath-1

a. 4
b. 6
c. 16
d. 64

4 control bits means the MUX can choose between 2^4=16 possible inputs. Each input has 6 wires.

20. If we were to replace the sign-extend module with a "zero-extend", how would this affect I-type instructions?

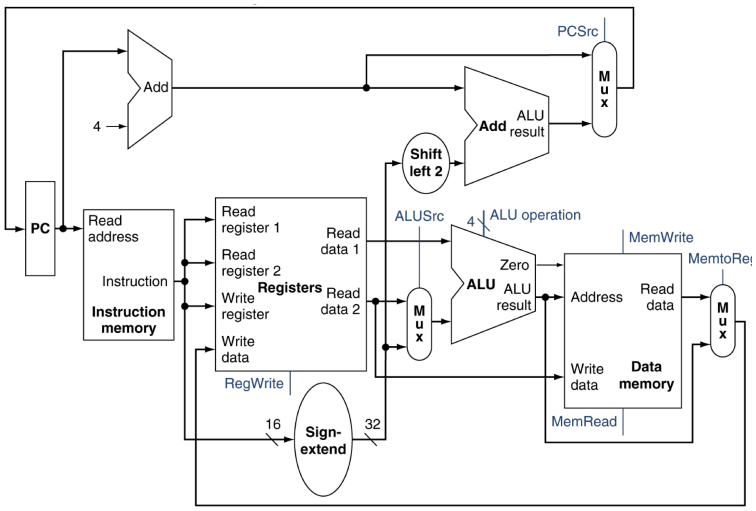


2015-Oct-Processor Control and Datapath-2

a. Not at all, the sign-extend is not used for I-type instructions
b. Not at all, the operand is already two's complement encoded
c. If the operand value is negative, then it changes value
d. If the operand value is positive, then it will become negative

Adding 0 to msb of a negative turns it into a different, positive value. Positive values are not affected.

21. There is a hardware error in our processor chip. As a result none of the I-type instructions work as they should (the other types work fine). Where could the error be?



2015-Oct-Processor Control and Datapath-3

a. The ALU mux input control signal (ALUSrc) is stuck at zero (always chooses Reg File input)
b. The sign-extend module sometimes flips LSB, so that a 1 becomes 0, and vice versa
c. MemToRg mux input control signal is stuck at one (always choose ALU output)
d. All of the above faults could be responsible for the error

All I-type instructions require the data from the sign-extended immediate field. Changing this MUX would prevent that data from getting to the ALU.

22. Which of these statements is NOT true?

2015-Oct-Processor Control and Datapath-4

a. There are no hazards in a single-cycle processor
b. Single-cycle processors have no use for a branch predictor
c. The ALU generally does not need a clock input
d. Clock cycle time for a single-cycle processor is longer than in a pipelined processor

Answer: D

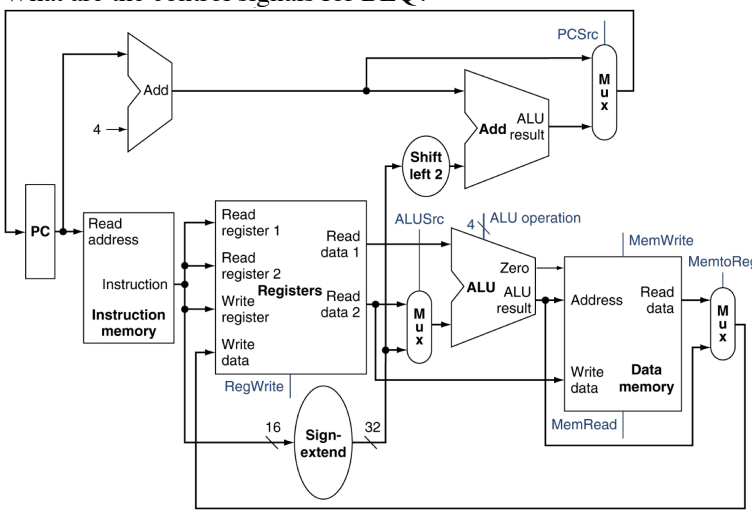
23. A single-cycle processor has three types of instructions: A takes 1ns, B takes 2ns, and C takes 10ns. A program with an instruction mix of 25% A, 70% B, and 5% C will run at what clock speed?

2015-Oct-Processor Control and Datapath-5

a. 1000 MHz (1/1ns)
b. 465 MHz (1/2.15ns)
c. 333 MHz (1/3ns)
d. 100 MHz (1/10ns)

The slowest instruction path dictates the clock speed.

24. What are the control signals for BEQ?



2015-Oct-Processor Control and Datapath-6

a. RegWrite=0, ALUSrc=Read Data 2, ALUOp=Sub, PCSrc=Zero
b. RegWrite=0, ALUSrc=SignExtend, ALUOp=Sub, PCSrc=!Zero

c. RegWrite=1, ALUSrc=Read Data 2", ALUOp=Sub, PCSrc=!Zero
d. RegWrite=0, ALUSrc=SignExtend, ALUOp=Add, PCSrc=Zero

BEQ does a subtraction so it needs to take data from the register file for both inputs and does not write back to the register file. The PC src comes from the Zero output of the ALU to branch if the results are equal.

25. Which of these instruction pairs have the same ALU operation?
`2018-Oct-Processor Control and Datapath-1`
    a. lw and add
    b. lw and bne
    c. add and bne
    d. sub and lw

lw does an add to generate the address, bne does a subtract to compare for zero. So lw and add will do the same operation (or bne and sub).

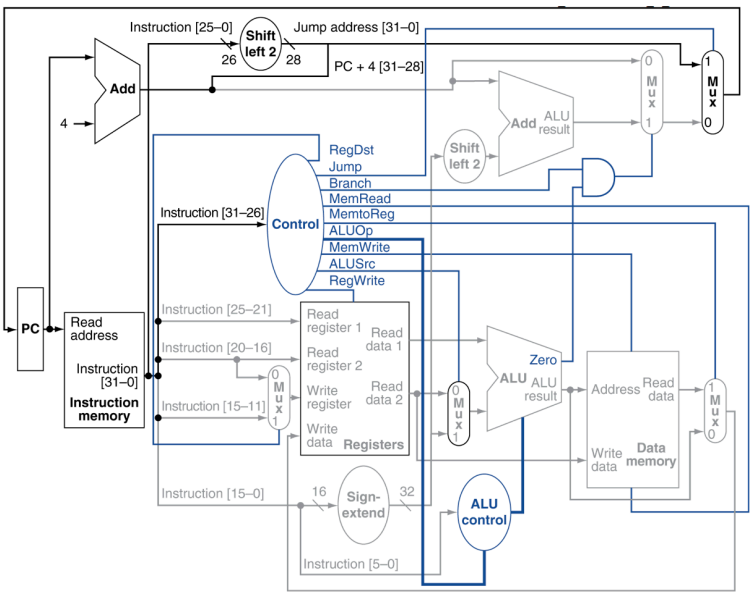
26. What registers do store instructions need to specify?
`2016-Aug-Processor Control and Datapath-1`
    a. One register for the address
    b. Two registers for address and data
    c. Three registers for address, data and writing into the register file
    d. None of the above

Only the address and data information is needed. Stores do not write back data values.

27. Why is a single "branch" output from the control logic not enough to handle both bne and beq instructions?
`2016-Aug-Processor Control and Datapath-2`
    a. We need to know which one looks at the ALU's Zero output
    b. We need to interpret the ALU's Zero output differently for each
    c. We do different calculations in the ALU for each
    d. All of the above

bne and beq interpret the ALU's Zero output differently. A Zero output true means the values are the same, so it is a branch for a beq but not for a bne. In both cases the ALU operation is the same.

28. What control flow instructions does this datapath support?



`2016-Aug-Processor Control and Datapath-3`
    a. BEQ, BNE
    b. J, BNE
    c. J, BEQ
    d. J, BEQ, BNE

The AND gate will only allow us to take the PC+Immediate branch if ALU Zero is true, which means the two values

are the same. That is, it only supports BEQ. We support Jump through the top path.

29. JAL is a J format instruction. How does the datapath know which register to store the return address into?
`2016-Aug-Processor Control and Datapath-4`
    a. The instruction specifies the destination register
    b. The immediate field specifies the destination for storing the return address
    c. The return address is automatically put in the PC register
    d. The value is generated by the control logic based on the opcode

J format instructions do not have space for anything other than a large constant and the opcode. The destination register needs to be generated by the control logic when it sees that the instruction opcode is JAL.

30. Why do we not have hazards in the single-cycle processor?
`2016-Aug-Processor Control and Datapath-5`
    a. Every instruction finishes before the next one starts so the register file always has the right data
    b. No instructions are executing in parallel so there are no conflicts between instructions
    c. Each instruction has access to the whole processor while it is executing so there is no fighting for resources
    d. All of the above

The pipelined register has problems because these things are not true!

31. How closely does the single-cycle datapath respect the ISA's promise of sequential and atomic execution?
`2016-Aug-Processor Control and Datapath-6`
    a. Sequential, but NOT atomic
    b. NOT sequential, but atomic
    c. BOTH Sequential AND atomic
    d. NEITHER sequential NOR atomic

The single-cycle datapath does the next instruction at a time and finishes it before any other instructions start. This means it is both sequential and atomic.

32. How closely does the single cycle datapath respect the ISA's promise of sequential and atomic execution?
`2014-Dec-Processor Control and Datapath-1`
    a. Sequential, but NOT atomic
    b. NOT sequential, but atomic
    c. BOTH Sequential AND atomic
    d. NEITHER sequential NOR atomic

The single-cycle datapath does the next instruction at a time and finishes it before any other instructions start. This means it is both sequential and atomic.

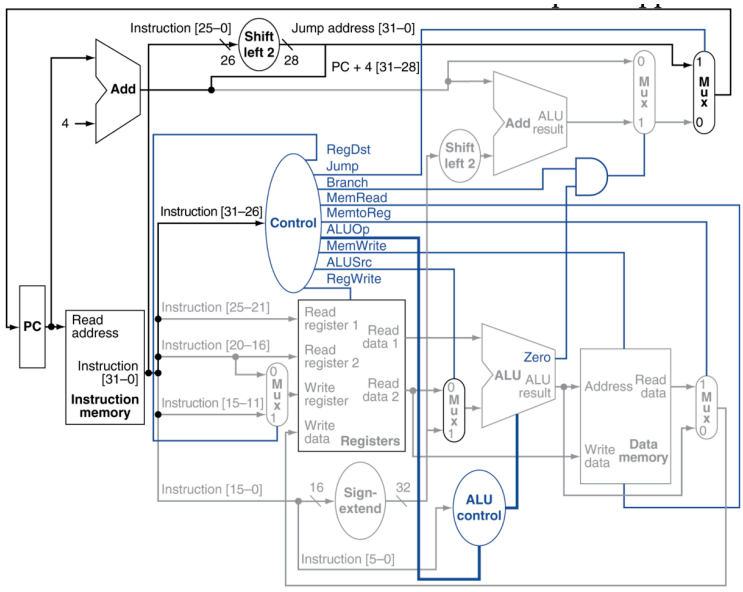
33. JAL is a J format instruction. How does the datapath know which register to store the return address into?
`2014-Dec-Processor Control and Datapath-2`
    a. The instruction specifies the destination register
    b. The immediate field specifies the destination for storing the return address
    c. The return address is automatically put in the PC register
    d. The value is generated by the control logic based on the opcode

J format instructions do not have space for anything other than a large constant and the opcode. The destination register needs to be generated by the control logic when it sees that the instruction opcode is JAL.

34. Which of the following statements is true about the value of the sign extender when a non I-format instruction is executed?
`2014-Dec-Processor Control and Datapath-3`
    a. The output is undefined, as there is not an immediate. The logic has invalid values as inputs and outputs.
    b. The sign-extension logic is turned off with for non I-format instructions.
    c. There is no immediate to sign-extend so the output is zero
    d. The sign extender extends bits from the instruction and produces garbage.

The sign extender always takes 15 bits from the instruction and sign-extends them. For instructions that don't define immediate, these bits are from the instruction and are therefore garbage.

35. What control flow instructions does this datapath support?

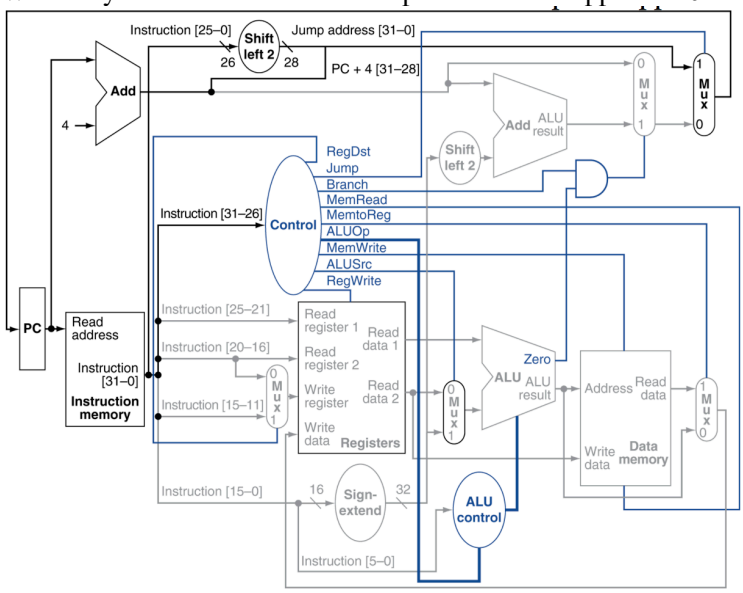


2014-Dec-Processor Control and Datapath-4

a. BEQ, BNE
b. J, BNE
c. J, BEQ
d. J, BEQ, BNE

The AND gate will only allow us to take the PC+Immediate branch if ALU Zero is true, which means the two values are the same. That is, it only supports BEQ. We support Jump through the top path.

36. What do you need to add to the datapath below to support the JR instruction?

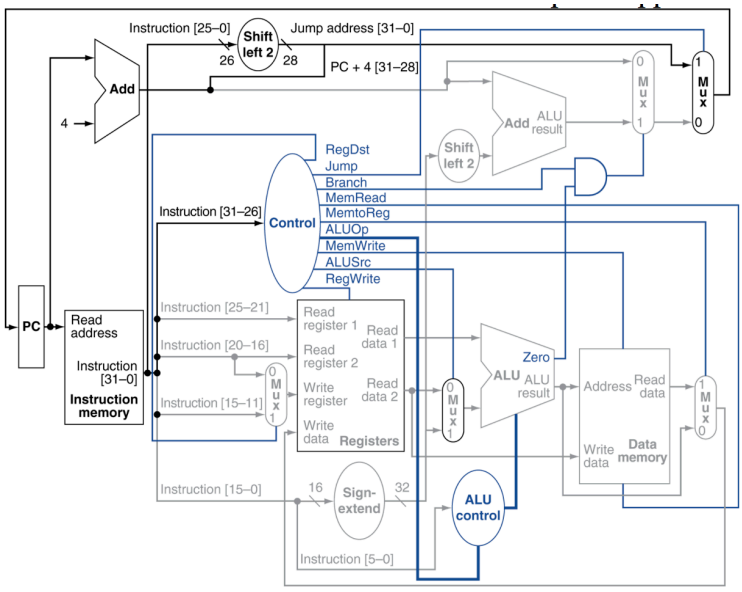


2014-Dec-Processor Control and Datapath-5

a. A MUX and control logic to send the PC to the register file
b. A MUX and control logic to send the register file data to the next PC
c. Both A and B
d. Nothing

For JR we need to take the PC from the register file and send it to the next PC, so we need a MUX to allow us to send the output of the register file to the next PC. For JAL we need the ability to send the PC into the register file.

37. What are the control signals in the datapath below for a beq instruction?

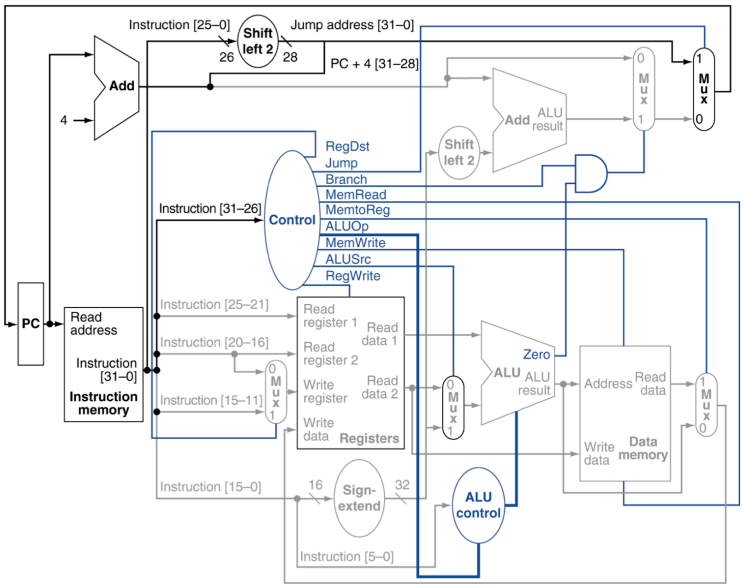


2014-Dec-Processor Control and Datapath-6

a. Branch=1, ALUSrc=0, ALUOp=SUB, RegWrite=0, Jump=0
b. Branch=1, ALUSrc=0, ALUOp=SUB, RegWrite=1, Jump=0
c. Branch=1, ALUSrc=1, ALUOp=SUB, RegWrite=0, Jump=0
d. Branch=1, ALUSrc=1, ALUOp=SUB, RegWrite=0, Jump=1

We need Branch=1 to do a branch, ALUSrc comes from the register file (0) and ALUOp is SUB so we calculate the difference. We don't write to the register file so RegWrite=0 and we are not doing a jump, so Jump=0.

38. Which part of the processor is storing state?



2015-Apr-Processor Control and Datapath-1

a. PC, registers, ALU
b. PC, registers, ALU, instruction memory
c. PC, registers, ALU, instruction memory, sign extension
d. PC, registers, instruction memory

The ALU is just combinational logic. It takes whatever its inputs are and generates outputs. The other components all store values and update them on the clock signal.

39. Why did our single-cycle processor design have two memories?

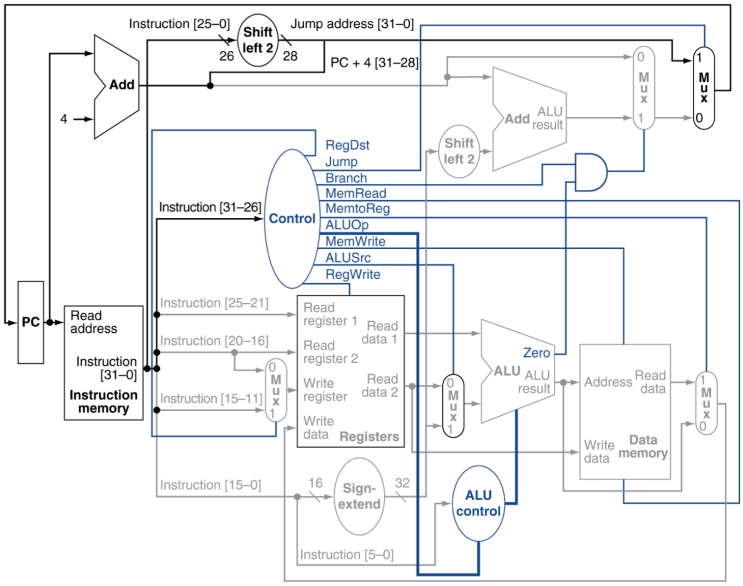
2015-Apr-Processor Control and Datapath-2

a. It doesn't, we just treat them as two different memories to understand better
b. Different parts of the instruction access the instruction memory and the data memory
c. Some instructions need to access the instruction memory and the data memory at the same time
d. We don't have a cache so it is faster if we have two memories

Load/store instructions need to both load the instruction (from the instruction memory) and access data (from the

data memory) at the same time.

40. What do we need to add to the processor shown below to support the jr instruction?

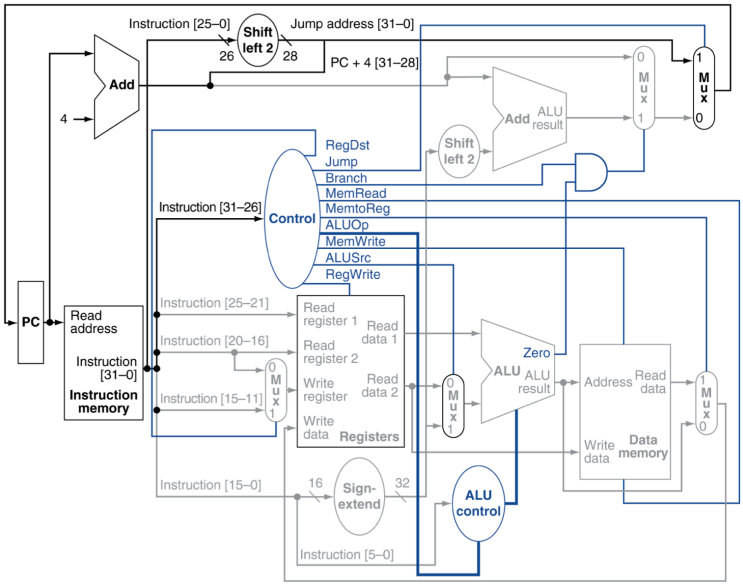


2015-Apr-Processor Control and Datapath-3

a. Another read and another write inputs to the Register File
b. Path from the PC to the Register File and another ALU function to the ALU control
c. Path from the Register File to the PC
d. All of the above

We need to be able to send the PC that was stored with the jal instruction in the register file back to the PC.

41. Which type of branch and jump instructions are supported by the datapath shown below?

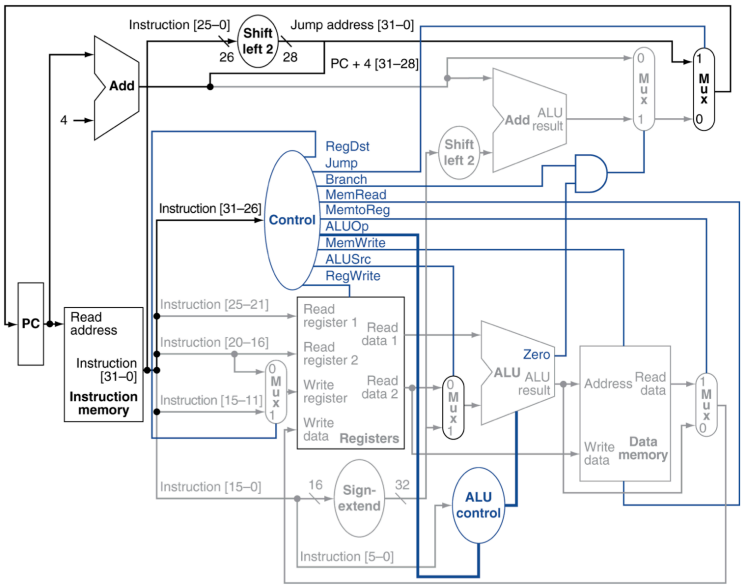


2015-Apr-Processor Control and Datapath-4

a. j
b. beq, j
c. bne, j
d. beq, bne, j

The datapath supports Branch AND Zero, which is branch if equal (beq) and can take bits directly from the instruction to do a jump.

42. What would happen if the RegDst MUX was removed from the datapath and it's 1 input was wired in directly?



2015-Apr-Processor Control and Datapath-5

a. No instructions could write back to the register file
b. I-type instructions could not write back to the register file
c. I-type instructions would write back to the wrong register
d. The processor would be slower due to hazards

The RegDst MUX is used to chose the right destination register bits to specify which register to write into. This is needed because I instructions use different bits for their destination than R instructions. If this was removed, the I instructions would end up writing to registers defined by some of the bits in their immediate field.

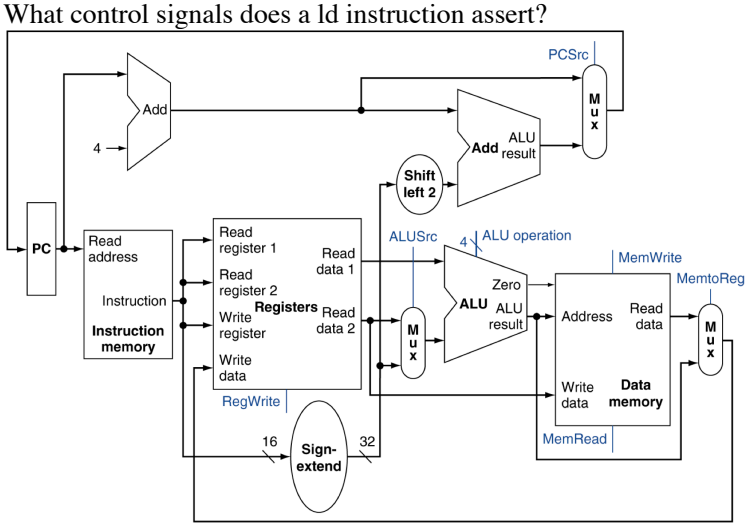
43. Why do we not have hazards in the single-cycle processor?

2015-Apr-Processor Control and Datapath-6

a. Every instruction finishes before the next one starts so the register file always has the right data
b. No instructions are executing in parallel so there are no conflicts between instructions
c. Each instruction has access to the whole processor while it is executing so there is no fighting for resources
d. All of the above

The pipelined register has problems because these things are not true!

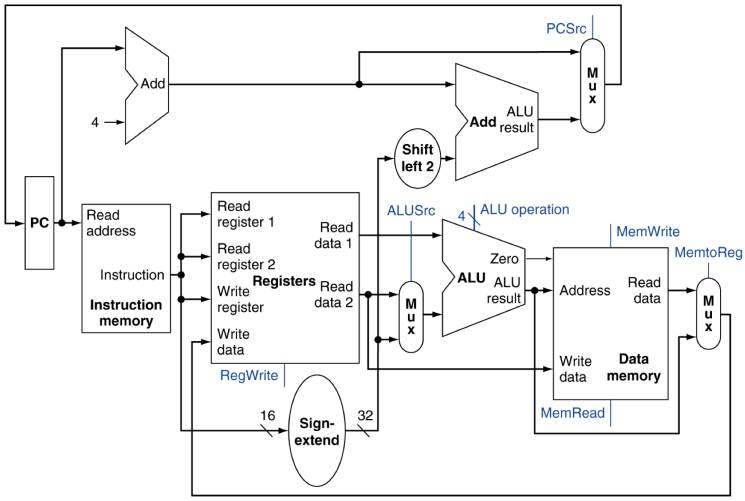
44. What control signals does a ld instruction assert?



2013-Jan-Processor Control and Datapath-1

a. RegWrite = 1, MemRead = 1, MemWrite = 0, ALUOp = None
b. RegWrite = 1, MemRead = 1, MemWrite = 0, ALUOp = Add
c. RegWrite = 0, MemRead = 1, MemWrite = 0, ALUOp = Add
d. RegWrite = 1, MemRead = 1, MemWrite = 1, ALUOp = Add

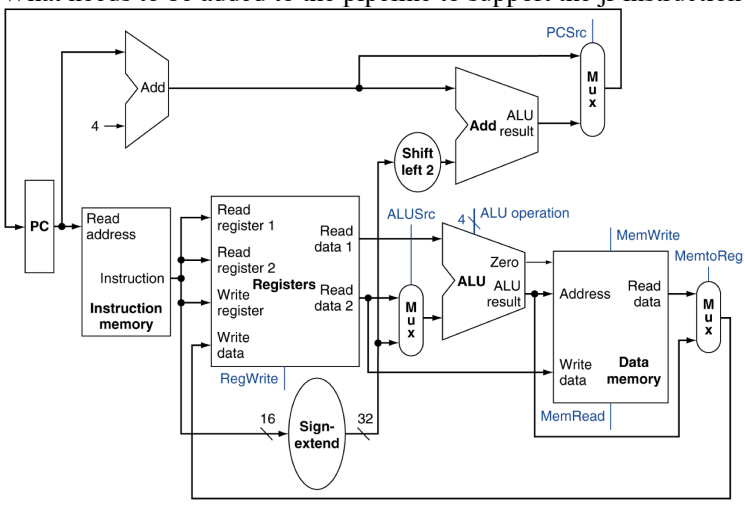
45. What needs to be done to determine if PCSrc should be set to the ALU result?



2013-Jan-Processor Control and Datapath-2

a. ALU=Sub, check zero, check conditional branch type
b. ALU=Add, check zero, check jump type
c. Check zero and branch type
d. Always set if beq

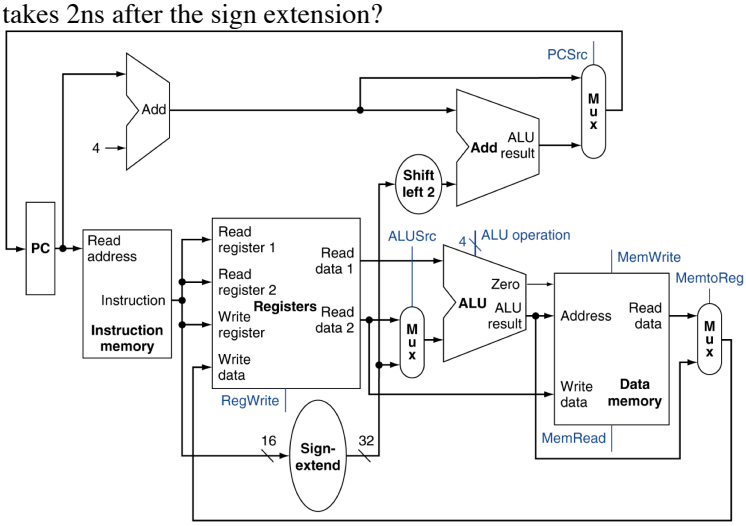
46. What needs to be added to the pipeline to support the jr instruction?



2013-Jan-Processor Control and Datapath-3

a. Path from the Register File to the PC
b. Path from the PC to the Register File
c. Another write input to the Register File
d. All of the above

47. What is the shortest clock cycle time for the pipeline below if the instruction memory takes 5ns, the registers take 1ns to read or write, sign-extension takes 0.5ns, the ALU takes 2ns, the data memory takes 10ns, and the branch computation takes 2ns after the sign extension?



2013-Jan-Processor Control and Datapath-4

a. 8ns
b. 18ns
c. 19ns
d. 21ns

48. Which element in the processor is not a state element?
`2013-Jan-Processor Control and Datapath-5`
    a. Instruction memory
    b. Registers
    c. ALU
    d. Data memory

49. Why do we not have hazards in the single-cycle processor?
`2013-Jan-Processor Control and Datapath-6`
    a. Every instruction finishes before the next one starts so the register file always has the right data
    b. No instructions are executing in parallel so there are no conflicts between instructions
    c. Each instruction has access to the whole processor while it is executing so there is no fighting for resources
    d. All of the above

50. How closely does the single-cycle datapath respect the ISA's promise of sequential and atomic execution?
`2015-Aug-Processor Control and Datapath-1`
    a. Sequential, but NOT atomic
    b. NOT sequential, but atomic
    c. BOTH sequential AND atomic
    d. NEITHER sequential NOR atomic

    The single-cycle datapath does the next instruction at a time and finishes it before any other instructions start. This means it is both sequential and atomic.

51. Why do we not have hazards in the single-cycle processor?
`2015-Aug-Processor Control and Datapath-2`
    a. Every instruction finishes before the next one starts so the register file always has the right data
    b. No instructions are executing in parallel so there are no conflicts between instructions
    c. Each instruction has access to the whole processor while it is executing so there is no fighting for resources
    d. All of the above

    The pipelined register has problems because these things are not true!

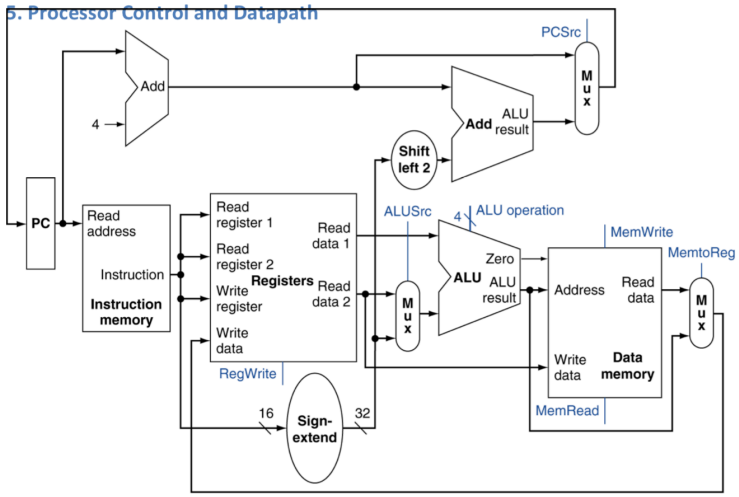
52. What are the control signals in the above datapath for a lw instruction?
`2015-Aug-Processor Control and Datapath-3`
    a. ALUSrc=SignExt, ALUOp=Add, RegWrite=1, MemRead=1, PCSrc=PC+4
    b. ALUSrc=SignExt, ALUOp=Add, RegWrite=1, MemRead=1, PCSrc=ALU
    c. ALUSrc=SignExt, ALUOp=NOP, RegWrite =1, MemRead=1, PCSrc=PC+4
    d. ALUSrc=Reg, ALUOp=Add, RegWrite =1, MemRead=1, PCSrc=PC+4

    The ALU needs to take a register and the sign-extended constant to generate the address (ALUSrc=SignExt, ALUOp=Add) and we need to write the results back to the register file (RegWrite=1) and tell the memory to read (MemRead=1). The next instruction is just PC+4, so PCSrc is PC+4.

53. Why do we need a MUX going into Write register in a full design?
`2015-Aug-Processor Control and Datapath-4`
    a. To choose the data to write into the register file
    b. To choose whether we decode an R- or I-format instruction
    c. To choose whether the destination is in the rt or rd position
    d. To choose which register we write to for all instructions

    For I-format instructions we need to take the destination register from the Rt field, so we need a MUX that chooses which instruction bits go into the write register address depending on the instruction type.

54. Which instructions are not supported on the datapath design below?

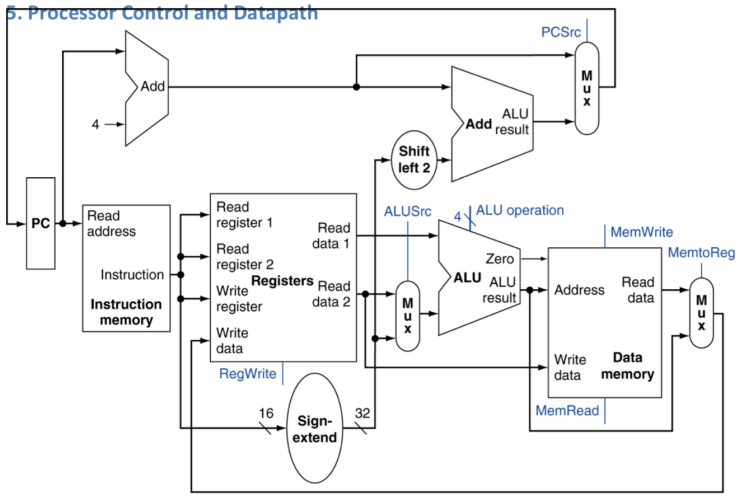


2015-Aug-Processor Control and Datapath-5

a. j
b. jal
c. jr
d. <u>All of the above</u>

The datapath above has no way for the PC to get to or from the register file, so it cannot support jal and jr. It also has no path for the larger constant from the j instruction to get to the PC, so it cannot support j.

55. What do you not need to know to determine PCSrc in the datapath design below?



2015-Aug-Processor Control and Datapath-6

a. ALU Zero out
b. The instruction
c. <u>ALUSrc</u>
d. You need al three

You need the Instruction to know if you are executing a bne or a beq. You then need ALU Zero out to determine if you should branch, but what you do depends on which instruction you have. ALUSrc is set by the type of instruction, so while it needs to choose the sign-extension for bne/beq, it will do that for all I-format instructions.

56. Why is it acceptable (performance-wise) to be able to do two reads and one write at the same time to the register file, but the memory can only do one read or one write at once?

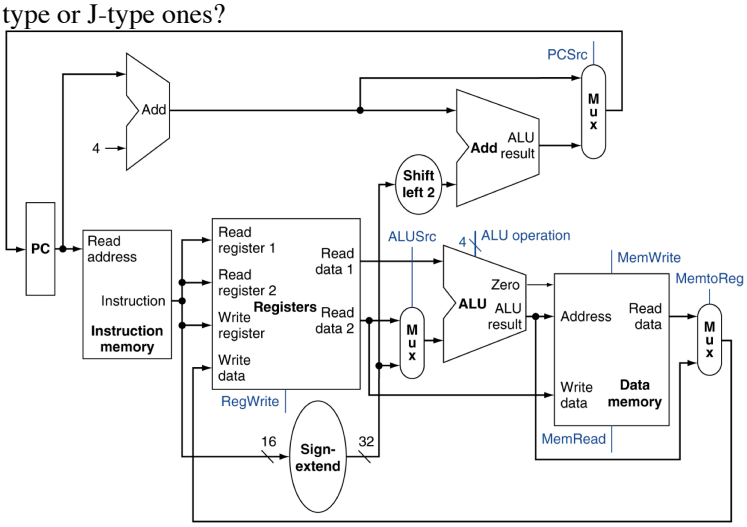
2014-Sep-Processor Control and Datapath-1

a. The memory is not used by most instructions, so it is okay to share the read and write ports
b. The register file is small, so the complexity of having many read and write ports is manageable
c. The memory is not read and written by the same instruction, so it is okay to share the read and write ports
d. <u>All of the above</u>

The register file is small so we can make it much more complex at an acceptable cost in size, power, and speed. The memory is too large for such a complex design, but since most operations are on data in the register file this is okay.

57. Which MUX would you disable (that is, permanently set to only one input) to break all I-type instructions but not affect R-

type or J-type ones?

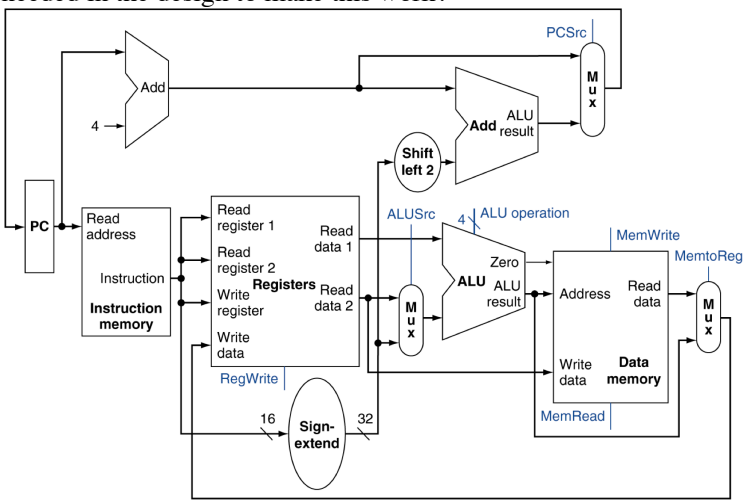


2014-Sep-Processor Control and Datapath-2

a. The top MUX
b. <u>The middle MUX</u>
c. The right MUX
d. Disabling one MUX is not enough

Disabling the middle MUX by setting it to only take the Register File Read data 2 output would prevent the immediate value from ever getting to the ALU, which would prevent all I-type instructions from working, while allowing others to work.

58. The R-format instructions use the third register address to determine where to write their results into the register file. The I-format instructions use the second register address to determine where to write their results into the register file. What is needed in the design to make this work?



2014-Sep-Processor Control and Datapath-3

a. Nothing. Just send the register address bits to the Write register input and the correct register will be selected for each instruction type.
b. A DEMUX to send the write data bus to the correct register depending on the instruction type.
c. <u>A MUX to choose which register address bits should be sent into the write register input depending on the instruction type.</u>
d. Control logic to delay the RegWrite signal on I-type instructions to make sure the data is in the right place.

A MUX is needed that selects the right bits from the instruction to use as the Write data address depending on the instruction type.

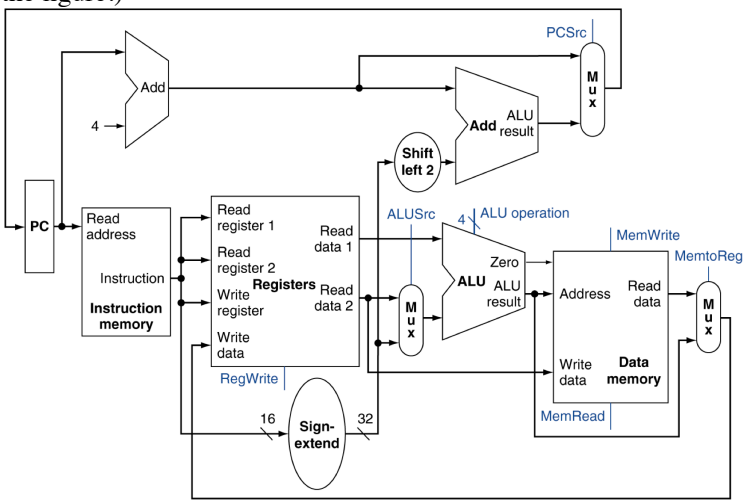
59. What is the logic that determines whether PCSrc should select PC+4 (0) or the ALU result (1)?

2014-Sep-Processor Control and Datapath-4

a. (Zero AND bne) OR (Zero AND beq)
b. <u>(!Zero AND bne) OR (Zero AND beq)</u>
c. (Zero AND bne) OR (!Zero AND beq)
d. (!Zero AND bne) OR (!Zero AND beq)

The ALU result should be used if we want to branch. If the instruction is beq this means Zero is true and if it is bne this means Zero is false.

60. Which instruction(s) could not be supported by the datapath as shown below. (Note that the control logic is not shown in the figure.)

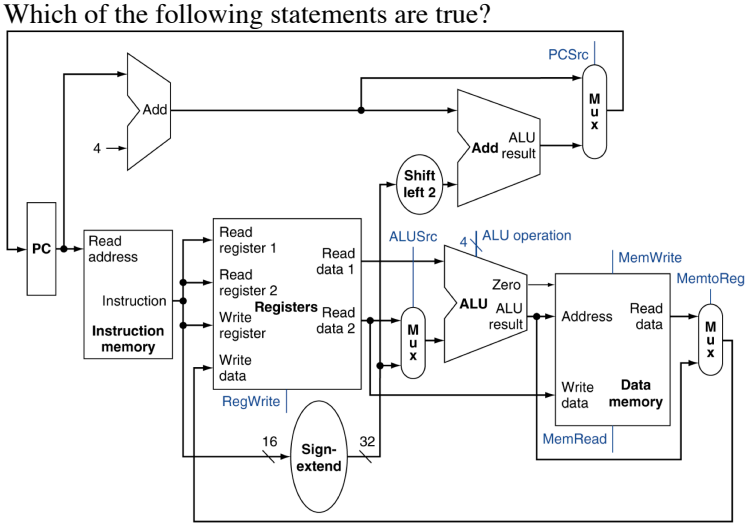


2014-Sep-Processor Control and Datapath-5

a. jr
b. bne
c. A and B
d. Both of the above are supported

bne can be supported since there is a path for the immediate to get to the PC. jr requires that the output of the register file get to the PC and there is no way for that to happen in the figure.

61. Which of the following statements are true?

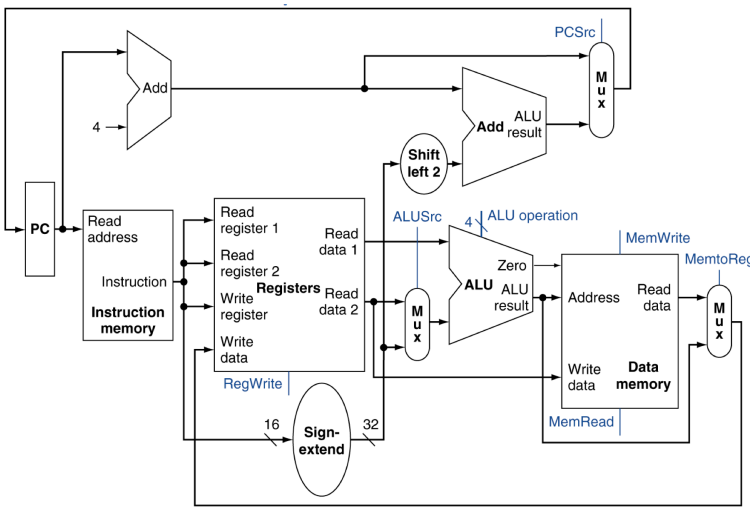


2014-Sep-Processor Control and Datapath-6

a. MemtoReg always selects the ALU result for R-type instructions
b. RegWrite is always true for I-type instructions
c. PCSrc always selects the ALU result for I-type instructions
d. All of the above

MemtoReg is always true for R-type instructions since they always write the ALU results back to a register. RegWrite is not true for st, which is an I-type instruction. PCSrc selects PC+4 for non branch instructions, which are I-type.

62. Which signal does sw not set?



2016-Oct-Processor Control and Datapath-1

a. MemWrite=1
b. ALUOp=Add
c. RegWrite=1
d. ALUSrc=SignExtend

Store word does not write into the register file so we do not set RegWrite.

63. A processor has a broken sign-extender that always outputs 0s in the upper 16 bits. How would this limit what it can do?

2016-Oct-Processor Control and Datapath-2

a. lw and sw would always use the same address
b. addi would not change the value
c. You could not load 32-bit constants
d. Conditional branches could only jump forwards

lw and sw would not add an offset, so they would use whatever address was in the register file. Addi would add only positive values, but it would still change the value. You could still load 32-bit constants by using lui and addi. However, branches would no longer be able to go backwards as 16-bit negative two's complement offsets would not be properly sign-extended. (The postivie ones would be.)

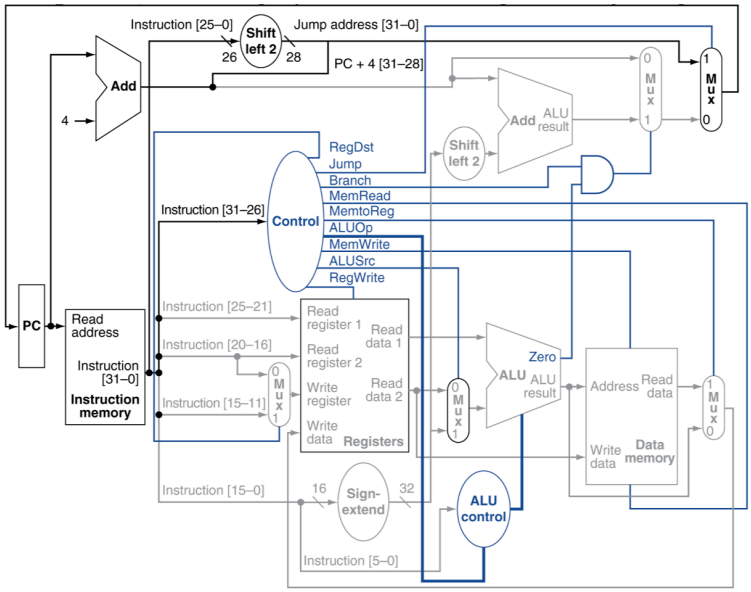
64. Why do we need two outputs from the control logic to properly handle bne and beq?

2016-Oct-Processor Control and Datapath-3

a. We need one for branch taken and one for not-taken
b. We need one for if Zero from the ALU is true and one for if it is false
c. We need two to determine how to use the Zero output for the two branches
d. We don't: we can do the logic with a single branch output from the control

We need to interpret the Zero output in the opposite manner if the branch type is bne or beq, so we need separate signals to know which is which..

65. Why do we have a MUX before the Write register going into the Register File in this datapath? (note it is grayed out in this figure, but just ignore that.)



2016-Oct-Processor Control and Datapath-4

a. To write PC+4 into the register file from JAL
b. To choose the write register address for I and R format
c. To choose the register to read out for JR
d. All of the above

The MUX chooses whether we use the 2nd or 3rd register field for the write destination so that the different I-type instructions can write to the correct register.

66. There are two binary control signals to choose the next PC in the above datapath: the output of the AND gate and the Jump output from the control. How many different next PC values do these two binary signals choose among?

2016-Oct-Processor Control and Datapath-5

a. 2
b. 3
c. 4
d. 232

While two binary signals can define 4 distinct choices, there are only 3 different branch addresses in this datapath: the beq from the ALU, jump from the shifter, and PC+4.

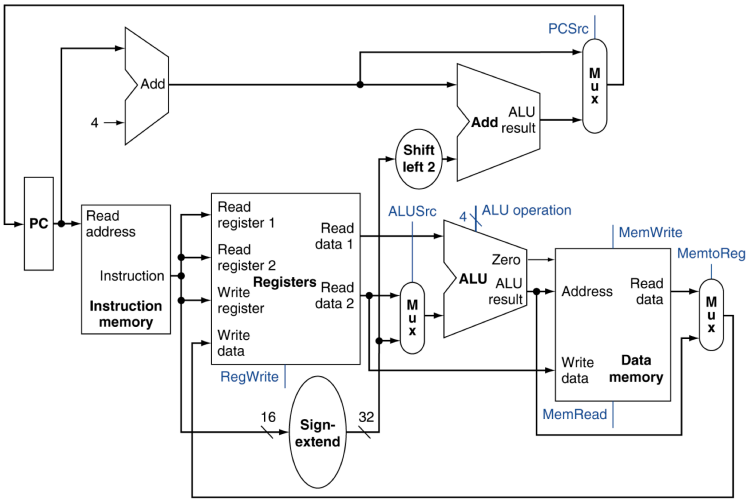
67. Why can we have two reads and one write at the same time from the register file, but we have to have separate instruction and data memories to read one instruction and one data word at the same time?

2016-Oct-Processor Control and Datapath-6

a. The instruction memory is not written so we can't read and write from it
b. The memory is not used by all instructions so we only access it sometimes
c. The register file is small, so we can build more logic to access it at a reasonable cost
d. All of the above

The register file is so small that we can afford to make it much more complex (reads and writes) without having it be too big or expensive. The memory is too large for that, but since most operations are in the register file it is okay.

68. What are the control signals for lw?

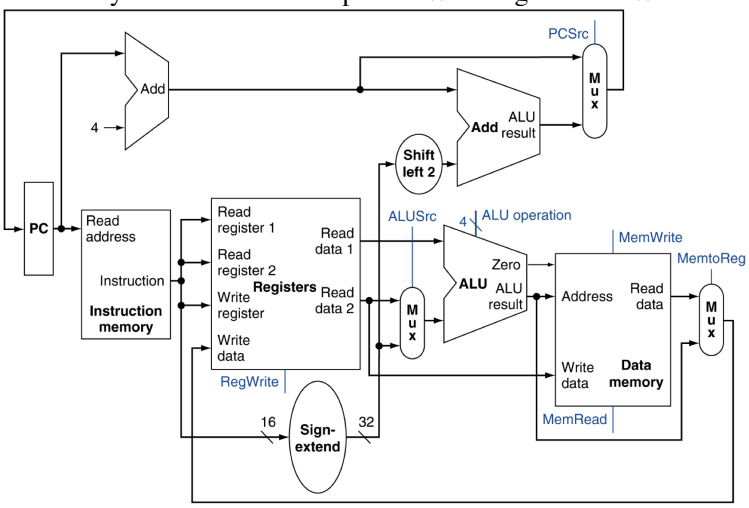


2016-Jan-Processor Control and Datapath-1

a. ALUOp=Add, ALUSrc=ReadData2, MemtoReg=Mem, RegWrite=1
b. ALUOp=Sub, ALUSrc= ReadData2, MemtoReg=Mem, RegWrite=1
c. ALUOp=Add, ALUSrc=SignExtend, MemtoReg=Mem, RegWrite=1
d. ALUOp=Sub, ALUSrc=SignExtend, MemtoReg=Mem, RegWrite=1

lw uses the ALU to compute the address by adding the SignExtended immediate to Register File output 1.

69. How many bits wide are the inputs to Write register and Write data in the datapath?



2016-Jan-Processor Control and Datapath-2

a. Write register=5, Write data=5
b. Write register=32, Write data=5
c. Write register=5, Write data=32
d. Write register=32, Write data=32

Write register selects the register to write to, and since there are 32 registers we need 5 bits. Write data is the full 32-bits of data we will write into the register file.

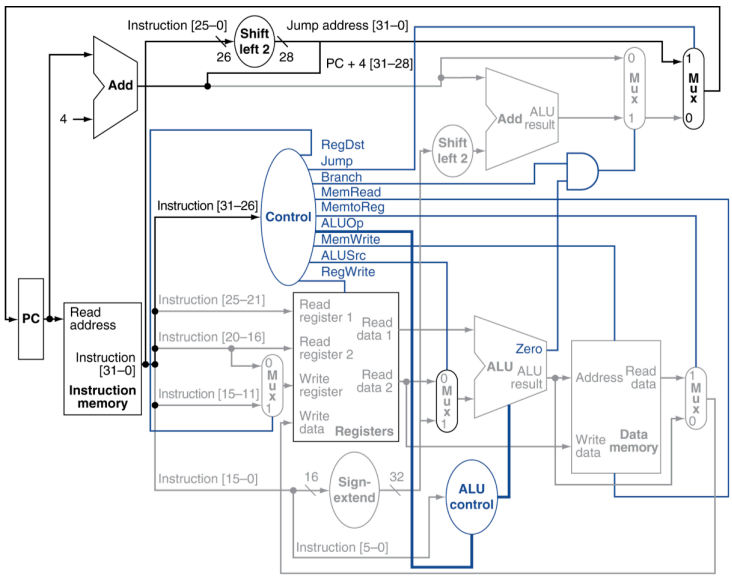
70. How much energy will the sign extender use when executing R-type instructions compared to I-type, and why?

2016-Jan-Processor Control and Datapath-3

a. None: R-type instructions don't have immediates
b. Same: R-type instructions have immediates but don't use them
c. Same: The sign extender does the same thing regardless of the instruction type
d. Less: R-type instructions don't have immediates

The sign extender always extends the immediate field, regardless of the instruction type. Therefore it will use the same amount of energy for all instruction types.

71. Which control flow instruction does this datapath support?



2016-Jan-Processor Control and Datapath-4

a. J
b. JR
c. JAL
d. BNE

This datapath has an AND gate for the Zero output which is only correct for BEQ. However, it also has a path from the instruction straight into the next PC, which is what you need for J.

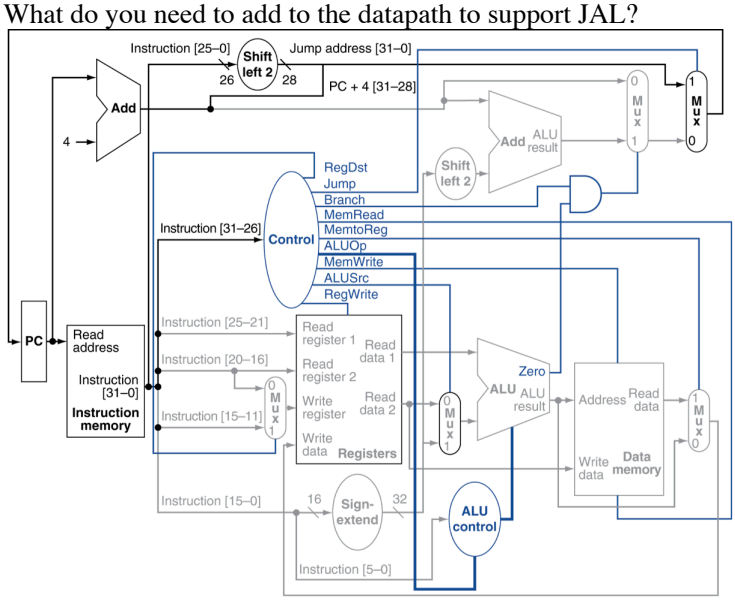
72. What logic is required to determine whether to use the branch address as the next PC for beq and bne?

2016-Jan-Processor Control and Datapath-5

a. Zero (just the Zero output from the ALU)
b. (Zero AND beq) OR (Zero AND bne)
c. (Zero AND beq) OR (!Zero AND bne)
d. (!Zero AND beq) OR (Zero AND bne)

You need separate signals from the control to know if the instruction is a beq or bne, because the branch condition is branch if beq and the registers are equal (Zero is true) or branch if bne and the registers are different (Zero is false).

73. What do you need to add to the datapath to support JAL?



2016-Jan-Processor Control and Datapath-6

a. Nothing
b. A connection from the next PC to the register file write data
c. Control logic and a MUX to write the next PC into the register file on JAL
d. B and C

JAL needs to get the next PC stored into the register file, which requires both a connection from the PC to the register file write data and the control logic to select it as appropriate.

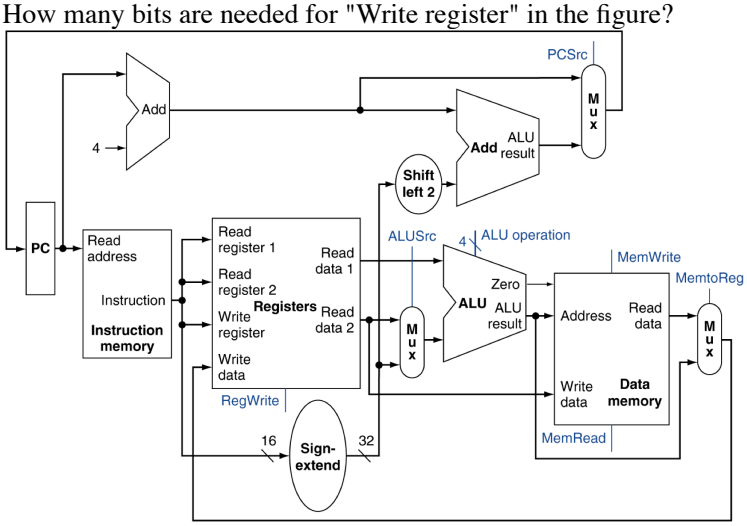
74. Why does our basic processor design have two memories?

`2013-Aug-Processor Control and Datapath-1`

a. It doesn't
b. We don't have a cache, so accessing the memory is slow
c. Different parts of the instruction access the instruction memory and the data memory
d. Some instructions need to access the instruction memory and the data memory at the same time

Load/store instructions need to both load the instruction (from the instruction memory) and access data (from the data memory) at the same time.

75. How many bits are needed for "Write register" in the figure?



`2013-Aug-Processor Control and Datapath-2`

a. 1
b. 5
c. 16
d. 32

5. This input chooses which register to write, so it needs 5 bits to choose from the 32 registers.

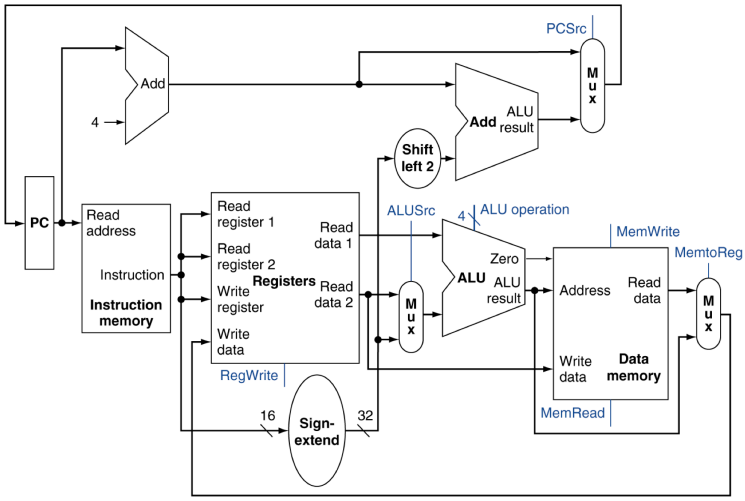
76. What is the output of the sign extender for non I-format instructions?

`2013-Aug-Processor Control and Datapath-3`

a. Zero: non-I-format instructions do not have an immediate to sign-extend
b. Nothing: non-I-format instructions do not have an immediate so the sign-extension logic is turned off
c. Garbage: non-I-format instructions do not have an immediate so the value is whatever bits were in that part of the instruction
d. Undefined: non-I-format instructions do not have an immediate so the sign-extension logic has unknown values coming into it and produces unknown values

Garbage. The sign extension logic sign extends the bits in the rd, shmt, and func fields of the instruction, no matter whether those are an immediate from an I-type instruction or the rd, shmt, and func fields for an R-type instruction. (the value is clearly defined, though.)

77. What determines the value of the ALUSrc MUX?

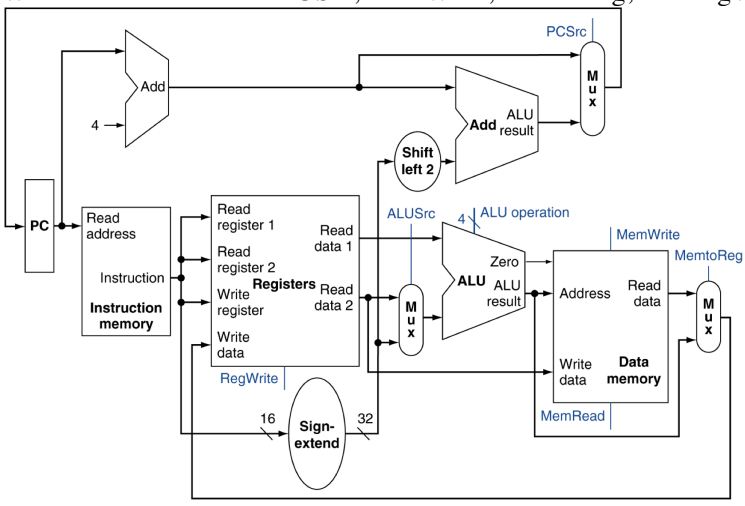


2013-Aug-Processor Control and Datapath-4

a. The instruction type: I=Read data 2, R=Sign-extend
b. The instruction type: I=Sign-extend, R=Read data 2
c. The ALU Zero output: True=Read data 2, False= Sign-extend
d. The value from Read data 2

The instruction type determines whether the ALU uses a second value from the register file (R-type) or the immediate field (I-type).

78. What are the values of ALUSrc, MemWrite, MemtoReg, and RegWrite for a lw instruction?



2013-Aug-Processor Control and Datapath-5

a. ALUSrc=Read-data-2; MemWrite=0; MemtoReg=Read-data; RegWrite=1
b. ALUSrc=Sign-extend; MemWrite=0; MemtoReg=Read-data; RegWrite=1
c. ALUSrc= Sign-extend; MemWrite=0; MemtoReg=ALU-result; RegWrite=1
d. None of these

The ALU calculates the address so it needs the immediate to add in (ALUSrc=Sign-extend); we are not writing to the memory with lw (MemWrite=0); the results are coming from the memory (MemtoReg=Read-data); and we do write the results into the register file (RegWrite=1)

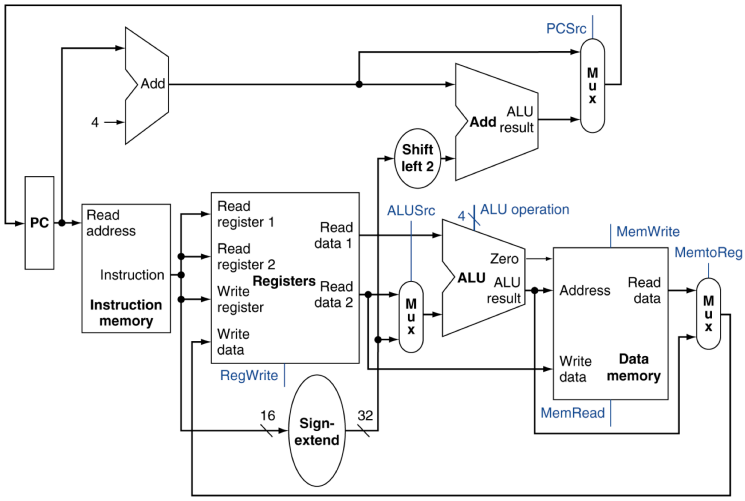
79. What is the ALU operation for a sw instruction?

2013-Aug-Processor Control and Datapath-6

a. Add
b. Subtract
c. Pass through
d. Undefined (ALU results not used)

The sw instruction needs to calculate the address by adding the immediate value to the register input, so the ALU does an ADD.

80. If the ALUSrc mux was removed, and Register Read data 2 was connected directly to the ALU, what instructions could no longer work?

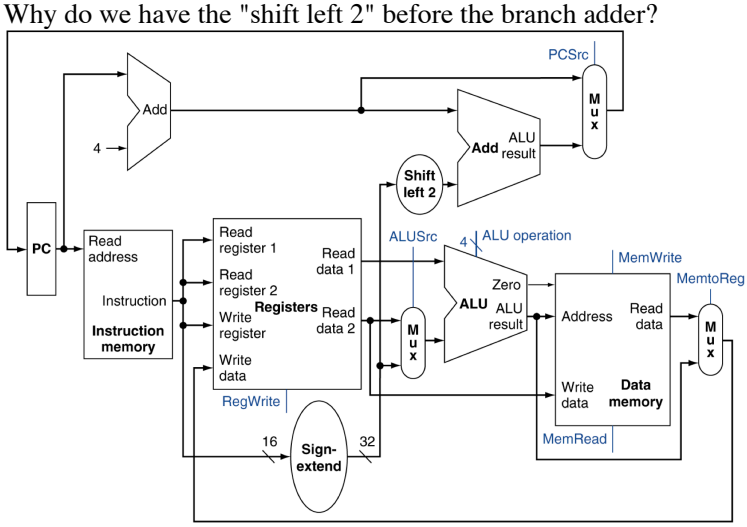


2013-Apr-Processor Control and Datapath-1

a. R-type
b. I-type
c. J-type
d. All of the above

I-type instructions require the data from the sign-extended immediate field. Changing this MUX would prevent that data from getting to the ALU.

81. Why do we have the "shift left 2" before the branch adder?



2013-Apr-Processor Control and Datapath-2

a. Because branch offsets are specified in bytes
b. Because branch offsets are specified in instructions
c. Because we don't want to change the last two bits of the instruction address
d. Because we can only jump in multiples of 4 instructions.

Branch offsets are specified in instructions, which are 4 bytes or a word. So we need to multiply the value by 4 (shift left 2) to get the address.

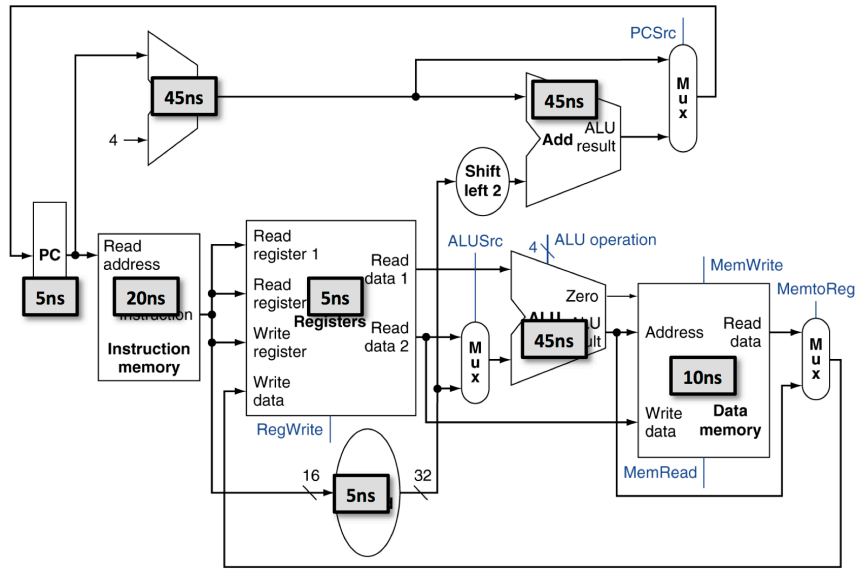
82. Which parts of the processor need a clock signal?

2013-Apr-Processor Control and Datapath-3

a. PC
b. Instruction memory
c. Register file
d. All of the above

The PC, instruction memory, and register file are all state elements, so they need a clock to know when to update.

83. What is the maximum clock speed of the processor below? (The times listed for the state elements are the time it takes to read or write.)



2013-Apr-Processor Control and Datapath-4

a. 85ns
b. 90ns
c. 100ns
d. 200ns

Longest path is the PC update due to the absurdly slow adders used there: 5+45+45+5 = 100.

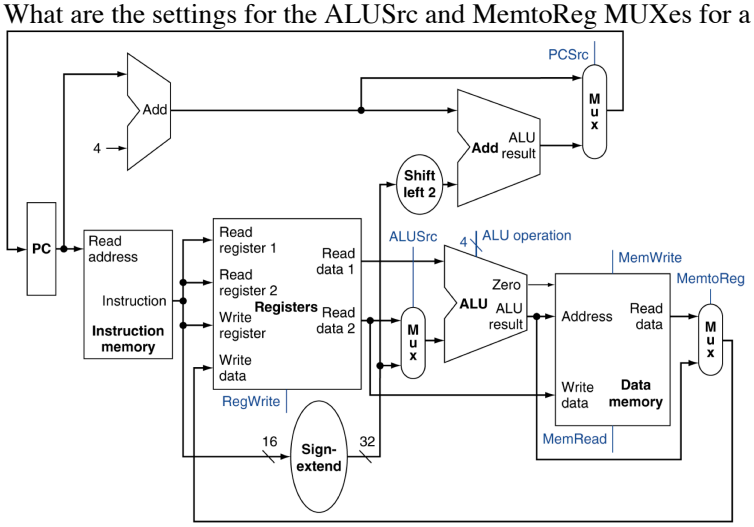
84. What is the ALU operation for a store word instruction?

2013-Apr-Processor Control and Datapath-5

a. Subtract
b. Add (from register file)
c. Add (from immediate)
d. Write to memory

The ALU does an add and it uses the immediate value because store word has an offset in its immediate field.

85. What are the settings for the ALUSrc and MemtoReg MUXes for a load word instruction?



2013-Apr-Processor Control and Datapath-6

a. ALUSrc = Read data 2, MemtoReg = Read data
b. ALUSrc = Read data 2, MemtoReg = ALU result
c. ALUSrc = Sign-extended, MemtoReg = Read data
d. ALUSrc = Sign-extended, MemtoReg = ALU result

We need the sign-extended immediate for the address offset (ALUSrc=Sign-extended) and we want the result from the memory (MemtoReg=Read data) since we are loading from memory.

6. Pipelining

1. Why do single-cycle processors have slow clock speeds?
   `2013-Dec-Pipelining-1`
   a. They have to handle a different length instructions
   b. They are limited by the slowest instruction
   c. They can't do any instructions in parallel
   d. All of the above

   While all of the above are true, it is only because they are limited by the slowest instruction that they have a slow clock speed. The others are complications (a) or performance problems (c) but they don't affect the clock speed directly.

2. What is the key thing that makes pipeline processors faster?
   `2013-Dec-Pipelining-2`
   a. Instructions are held in pipeline registers
   b. Instructions are executed in parallel
   c. Instructions can take different numbers of cycles
   d. All of the above

   Executing instructions in parallel is the key to pipeline performance. We use each part of the processor for a different instruction at the same time.

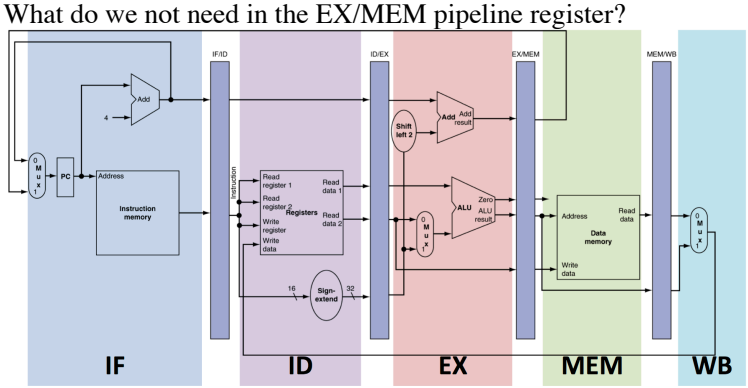
3. A single-cycle processor has a 200ns clock. It is pipelined to 10 stages with reasonable pipeline registers. What is the clock speed for the resulting processor? (Hint: think about what the likely clock will be if the designer decided to pipeline to 10 stages and what portion of that is likely to be used by the pipeline registers.)
   `2013-Dec-Pipelining-3`
   a. 20ns
   b. >20ns and <200ns
   c. 200ns
   d. >200ns

   The pipeline registers will take some time, but not as much as the whole processor (the "reasonable" bit). This means each stage will be 200ns/10=20ns+pipeline register time. So the clock will run slower than 20ns, but faster than the original processor.

4. What's the key thing for making a pipelined processor run fast?
   `2013-Dec-Pipelining-4`
   a. High clock speed
   b. Lots of pipeline stages
   c. Forwarding and branch prediction
   d. Keeping the pipeline full

   Keeping the pipeline full. Speed and stages aren't useful if the pipeline is often empty. Forwarding and branch prediction are techniques to keep the pipeline full.

5. What do we not need in the EX/MEM pipeline register?



   `2013-Dec-Pipelining-5`
   a. ALU result
   b. RF 2
   c. RF 1
   d. Branch address

   We need the ALU result to send to the memory or for write back. We need RF2 if we are going to do a write as that

is the data to write. We need the Branch address since the branch is done in the MEM stage. We do not need RF1 because it has already been used as an input to the ALU.

6. Really long pipelines could be great: if we split the instructions into 100 smaller chunks we could run at 100x the clock frequency. So why don't we do this?
`2013-Dec-Pipelining-6`
    a. Hard to break instructions down to really small pieces
    b. Pipeline register overhead will kill us
    c. Hard to keep long pipelines full
    d. All of the above

All of the above are true. The biggest problem, however, is keeping the pipeline full.

7. How much faster does a processor run if we divide each instruction into 8 cycles, run the clock 4 times faster, and only run one instruction at a time?
`2017-Jan-Pipelining-1`
    a. 32x faster
    b. 2x faster
    c. 2x slower
    d. 32x slower

Each instruction takes 8 cycles to finish, and each cycle is 1/4 as long, so it will take 8/4 or 2x longer to finish each instruction. We need to run them in parallel to get a pipelined benefit.

8. A processor is perfectly pipelined with 5 stages and runs at 5x the clock speed with no overhead, but the pipeline is only 80% full. What is the speedup over the original processor?
`2017-Jan-Pipelining-2`
    a. 0.8x
    b. 4x
    c. 5x
    d. 20x

If the pipeline was kept full we would finish instructions 5x faster. But since it's kept only 80% full, we get only 80% of this speedup, or 4x.

9. A manager analyzes the company's next generation processor design and realizes that the processor can be divided into 10 even stages and run at 10x the frequency to match that. The manager knows that the branch predictor is good enough to ensure that the pipeline will always be 90% full. The designer tells the manager that at that frequency the overhead of the pipeline latches will be 100% (e.g., each latch will take the same amount of time as the logic in the pipeline stage). What is the expected performance for this design?
`2017-Jan-Pipelining-3`
    a. 4.5x
    b. 9x
    c. 18x
    d. 20x

A perfectly pipelined processor with 10 stages should be 10x faster, but if it is only 90% full it should have been 9x faster. However, if the pipeline latches take up as much time as each stage, then half the time will be spent in the latches, so the performance will only be half, or 4.5x.

10. A single?cycle processor takes 100ns per instruction. It is perfectly pipelined into 10 stages with pipeline registers that take 2ns and can be kept 60% full. What change in instruction throughput can you expect compared to the single?cycle processor?
`2017-Jan-Pipelining-4`
    a. 60% slowdow
    b. 60% speedup
    c. 5x speedup
    d. 10x speedup

If perfectly pipelined the processor would have 10x the throughput, but each instruction takes 12ns (10 for the stage plus 2 for the pipeline register) so the pipeline will finish an instruction every 12ns, which is an increase of 100ns/12ns = 8.33x. However, if the pipeline is only 60% full, the throughput speedup will only be 60% of that, or 5x.

11. Why do we need to keep the RF2 data in the EX/MEM pipeline register?
`2017-Jan-Pipelining-5`
    a. For the ALU to handle R-type instructions
    b. For the memory data
    c. For the pipeline to write back to the register file
    d. We don't

    The RF2 output is the data we write to the memory in a store instruction. We need to keep it in the pipeline until we get to the MEM stage so we have the data to write to the memory.

12. A single-cycle processor takes 100ns and uses 100pJ. It is perfectly pipelined to 20 stages (20 registers), with pipeline registers that take 2ns and 2pJ in energy. What is the energy per instruction for the pipelined processor as a percentage of the original one?
`2017-Jan-Pipelining-6`
    a. 20%
    b. 40%
    c. 100%
    d. 140%

    20 stages will cost 100pJ to compute plus 20*2pJ in energy, or 140pJ. This is 140/100 = 140% of the energy per instruction.

13. What do you need to do to make sure a given pipeline can reach its full potential?
`2014-Apr-Pipelining-1`
    a. Lots of stages
    b. Even stages
    c. High clock frequencies
    d. Keeping the pipeline full

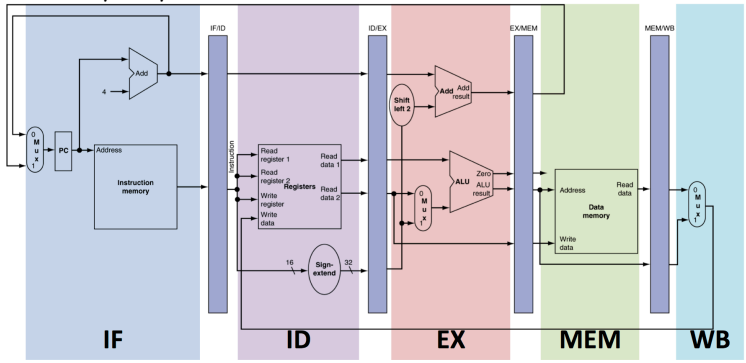
    The only way to get close to the potential of pipelining is to keep the pipeline full. This is made difficult by hazards and delay slots.

14. Does the pipelined processor execute instructions in an atomic and in-order manner?
`2014-Apr-Pipelining-2`
    a. Atomic: yes, In-order: yes
    b. Atomic: yes, In-order: no
    c. Atomic: no, In-order: yes
    d. Atomic: no, In-order: no

    The pipeline executes instructions in neither and atomic nor in-order manner. The instructions now happen in 5 separate pieces (not atomically) and the don't happen in-order (e.g., one instruction's EX may happen before an earlier instruction's MEM).

15. What value will be in R2 after this code executes on the pipeline shown below?

```
addi R2, R0, 4
add R3, R0, R4
beq R3, R4, skip
addi R2, R0, 0
skip:
addi R2, R2, 2
addi R2, R2, 4
```



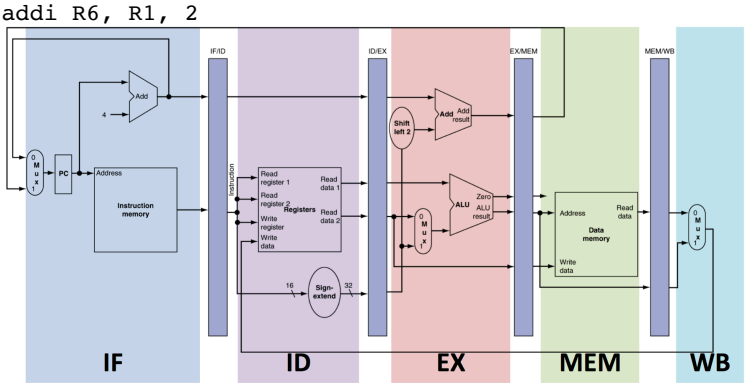
`2014-Apr-Pipelining-3`

a. 4
b. 6
c. 10
d. 12

R2: 4, then set to 0, then +2, then +4, then we resolve the branch, and we go back and do +2 and +4 again. So we end up with 0+2+4+2+4=12. Remember that with the pipeline above we have three branch delay slots.

16. How many NOPs need to be inserted in this code to make it execute correctly on the pipeline shown below? (You are allowed to re-arrange instructions.)

```
iterate:
sub R7, R7, -1
or R6, R6, R4
bne R4, R5, iterate
addi R5, R5, 1
addi R6, R1, 2
```



2014-Apr-Pipelining-4

a. 0
b. 1
c. 2
d. 3

The branch takes 3 cycles to resolve, so we need 3 NOPs unless we can move other instructions. The only instruction we can move is the OR instruction since the two addi instructions cannot execute if the branch is not taken. So we can cover one NOOP and we therefore only need 2 extras.

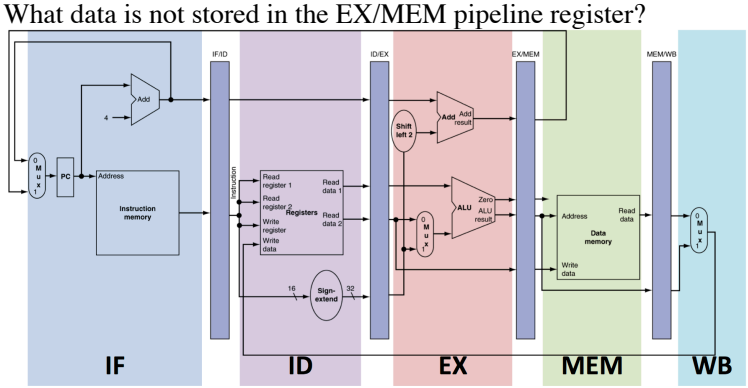
17. What is the speedup for a processor with a 200ns clock frequency that is equally pipelined to 25 stages with pipeline latches that take 2ns each?

2014-Apr-Pipelining-5

a. 10x
b. 17x
c. 20x
d. 25x

The clock frequency will now be 200/25=8 plus 2ns, or 10ns per stage. This means that if we keep the pipeline full it will run 200/10 or 20x faster.

18. What data is not stored in the EX/MEM pipeline register?



2014-Apr-Pipelining-6

a. Register File 1 output
b. Register File 2 output

c. ALU result
d. Branch address

We do not need to keep Register File 1 output in this stage since it has already been put into the ALU.

19. A processor is perfectly pipelined with 5 stages and runs at 5x the clock speed with no overhead, but the pipeline is only 80% full. What is the speedup over the original processor?
`2015-Oct-Pipelining-1`
    a. 0.8x
    b. 4x
    c. 5x
    d. 20x

If the pipeline was kept full we would finish instructions 5x faster. But since it's kept only 80% full, we get only 80% of this speedup, or 4x.

20. How does pipelining improve performance?
`2015-Oct-Pipelining-2`
    a. It divides instructions into smaller steps
    b. It allows the processor to work on multiple instructions at the same time
    c. It allows shorter instructions to finish sooner
    d. It reduces the instructions' latencies

Pipelining allows overlapping execution of multiple instructions with the same circuitry. The circuitry is usually divided up into stages, including instruction decoding, arithmetic, and register fetching stages, wherein each stage processes one instruction at a time.

21. What is the role of pipeline registers?
`2015-Oct-Pipelining-3`
    a. They store the results of each stage so that the next stage can use them
    b. They allow the register file to store more values
    c. They reduce the number of NOPs we need to insert in the pipeline when executing a branch instruction
    d. All of the above

Pipeline registers store partial results from each pipeline stage and make them available to the next stage at the next clock cycle.

22. A single-cycle processor takes 100ns per instruction. It is perfectly pipelined into 10 stages with pipeline registers that take 2ns and can be kept 60% full. What change in instruction throughput can you expect compared to the single-cycle processor?
`2015-Oct-Pipelining-4`
    a. 60% slowdow
    b. 60% speedup
    c. 5x speedup
    d. 10x speedup

If perfectly pipelined the processor would have 10x the throughput, but each instruction takes 12ns (10 for the stage plus 2 for the pipeline register) so the pipeline will finish an instruction every 12ns, which is an increase of 100ns/12ns = 8.33x. However, if the pipeline is only 60% full, the throughput speedup will only be 60% of that, or 5x.

23. A single-cycle processor takes 100ns per instruction. It is perfectly pipelined into 10 stages with pipeline registers that take 2ns and can be kept 60% full. What change in instruction latency can you expect compared to the single-cycle processor?
`2015-Oct-Pipelining-5`
    a. 20% speedup
    b. 20% slowdown
    c. 60% speedup
    d. 60% slowdown

Each stage takes 12ns (10 for the stage +2 for the pipeline register), and there are 10 of them, so an instruction will spend 120ns in total to complete, while it took 100ns on the single-cycle processor. This is a 20% slowdown in latency. The pipeline occupation plays no role in this..

24. What is the advantage of a multi-cycle CPU compared to a single-cycle CPU?
`2015-Oct-Pipelining-6`
    a. The clock speed is no longer limited by the slowest instruction path
    b. The CPU can analyze the lengths of the instructions currently in the pipeline and adjust its clock speed accordingly
    c. Each instruction can take a different number of cycles to execute
    d. A and C

    In a single-cycle CPU, the clock speed is limited by the slowest instruction path. In a multi-cycle CPU, each instruction can take a different amount of cycles, which means that "slow" instructions can use more than one cycles to complete.

25. What is a consequence of decreasing the number of stages in the pipeline for a processor that keeps the pipeline full?
`2018-Oct-Pipelining-1`
    a. The area required to build the processor would decrease
    b. The critical path length would decrease
    c. The throughput would increase
    d. The latency per instruction would increase

    With fewer stages, you will need fewer pipeline registers, which will decrease the overall area. The critical path length will increase because you are doing more in each stage. The throughput will decrease because you are unable to do as many instrucitons at the same time. The latency will decrease, as you will have fewer pipeline register delays for each completing instruction.

26. A new processor changes the previous single-cycle design in two ways: 1) it divides each instruction into 24 steps, and, 2) it runs the clock 24 times faster. When they test the processor they find it is not faster than the previous design. Which of the following is the most important change they need to make?
`2018-Oct-Pipelining-2`
    a. Keep the pipeline full
    b. Execute multiple instruction parts at the same time
    c. Forward for hazzards
    d. All are equally important

    To make a pipeline effective you need to execute the instruction steps in parallel! They just cut them up and did them one-after-another. Keeping the pipeline full and forwarding are only important once you have a pipeline.

27. What is the speedup if you have an ideally pipelined processor with 15 stages, but due to control hazards you can only keep it full of instructions 66% of the time?
`2016-Aug-Pipelining-1`
    a. 0.66x
    b. 10x
    c. 15x
    d. 22.7x

    An ideally pipelined 15-stage processor would be 15x faster, but if we can only keep it 66% full, then we will lose 33% of the performance, so we will be 10x faster .

28. A manager analyzes the company's next generation processor design and realizes that the processor can be divided into 10 even stages and run at 10x the frequency to match that. The manager knows that the branch predictor is good enough to ensure that the pipeline will always be 90% full. The designer tells the manager that at that frequency the overhead of the pipeline latches will be 100% (e.g., each latch will take the same amount of time as the logic in the pipeline stage). What is the expected performance for this design?
`2016-Aug-Pipelining-2`
    a. 4.5x
    b. 9x
    c. 18x
    d. 20x

    A perfectly pipelined processor with 10 stages should be 10x faster, but if it is only 90% full it should have been 9x faster. However, if the pipeline latches take up as much time as each stage, then half the time will be spent in the latches, so the performance will only be half, or 4.5x.

29. A single-cycle processor has a 200ns clock. It is pipelined to 10 stages with reasonable pipeline registers. What is the clock speed for the resulting processor

2016-Aug-Pipelining-3

a. 20ns
b. >20ns and <200ns
c. 200ns
d. >200ns

The pipeline registers will take some time, but not as much as the whole processor (the "reasonable" bit). This means each stage will be 200ns/10=20ns+pipeline register time. So the clock will run slower than 20ns, but faster than the original processor.

30. What is the advantage of a multi?cycle CPU compared to a single?cycle CPU?

2016-Aug-Pipelining-4

a. The clock speed is no longer limited by the slowest instruction path
b. The CPU can analyze the lengths of the instructions currently in the pipeline and adjust its clock speed accordingly
c. Each instruction can take a different number of cycles to execute
d. A and C

In a single-cycle CPU, the clock speed is limited by the slowest instruction path. In a multi-cycle CPU, each instruction can take a different amount of cycles, which means that "slow" instructions can use more than one cycles to complete.

31. What's the key trick to making pipelines fast?

2016-Aug-Pipelining-5

a. Lots of stages
b. Even stages
c. High clock frequencies
d. Keeping the pipeline full

The only way to get close to the potential of pipelining is to keep the pipeline full. This is made difficult by hazards and delay slots.

32. A single-cycle processor takes 100ns and uses 100pJ. It is perfectly pipelined to 20 stages (20 registers), with pipeline registers that take 2ns and 2pJ in energy. What is the energy per instruction for the pipelined processor as a percentage of the original one?

2016-Aug-Pipelining-6

a. 20%
b. 40%
c. 100%
d. 140%

20 stages will cost 100pJ to compute plus 20*2pJ in energy, or 140pJ. This is 140/100 = 140% of the energy per instruction.

33. Which of the following statements is not true about pipelined processors?

2014-Dec-Pipelining-1

a. Commercial processors today typically have 15-25 pipeline stages.
b. It is possible to design processors with multiple parallel pipelines of different lengths.
c. Doubling the pipeline length doubles the processing speed.
d. Processors use separate instruction and data caches so they can load an instruction at the same time as they load data.

If you make the pipeline longer it becomes increasingly harder to keep the pipeline full, so twice as long a pipeline will not run twice as fast.

34. What is the speedup you expect if you add 20 ideal pipeline stages to a processor, but you can only keep them full with instructions 25% of the time?

2014-Dec-Pipelining-2

a. 2.5x
b. 5x
c. 19.75x
d. Need more information

The ideal speedup for 20 pipeline stages is 20x (if the clock speed is increased accordingly). However, if we only can

keep them full of instructions 25% percent of the time, we have 1/4 of the benefit (0.25*20), or 5x.

35. An un-pipelined processor takes 250ns per instruction. Suppose you re-design it with 50 pipeline stages, where each stage needs a pipeline register that adds an overhead of 5ns. What would be the latency to complete each individual instruction on the new design compared to the un-pipelined one, if we assume the pipeline can be kept completely full?
`2014-Dec-Pipelining-3`
    a. 0.5 (lower latency)
    b. 2x
    c. 5x
    d. 250x

Each instruction needs 50 stages, where each stage takes 250ns/50 logic and 5ns because of the overhead. This means each stage takes 10ns, and if there are 50 stages then in total will take 500ns. Previously, each instruction took 250ns, so it is 2x larger

36. Why do we need to keep the RF 2 output data in both the ID/EX and EX/MEM pipeline registers?
`2014-Dec-Pipelining-4`
    a. We need to keep it through the pipeline to write back
    b. We may need it for the ALU or the branch address
    c. We may need it for the ALU or the memory data
    d. We don't

The RF 2 output is used for both ALU operations (R-format instructions that use two register file inputs) and for store operations as the data to store. We have to keep it in the ID/EX stage for the ALU and all the way to the EX/MEM stage in case we have a store instruction and need it for the memory.

37. A single-cycle processor takes 100ns and uses 100pJ. It is perfectly pipelined to 20 stages (20 registers), with pipeline registers that take 2ns and 2pJ in energy. What is the energy per instruction for the pipelined processor as a percentage of the original one?
`2014-Dec-Pipelining-5`
    a. 20%
    b. 40%
    c. 100%
    d. 140%

20 stages will cost 100pJ to compute plus 20*2pJ in energy, or 140pJ. This is 140/100 = 140% of the energy per instruction.

38. A single-cycle processor takes 100ns and uses 100pJ. It is perfectly pipelined to 20 stages (20 registers), with pipeline registers that take 2ns and 2pJ in energy. If the pipeline is 80% full, what is the throughput of the pipelined processor relative to the original one?
`2014-Dec-Pipelining-6`
    a. 11x
    b. 14x
    c. 16x
    d. 20x

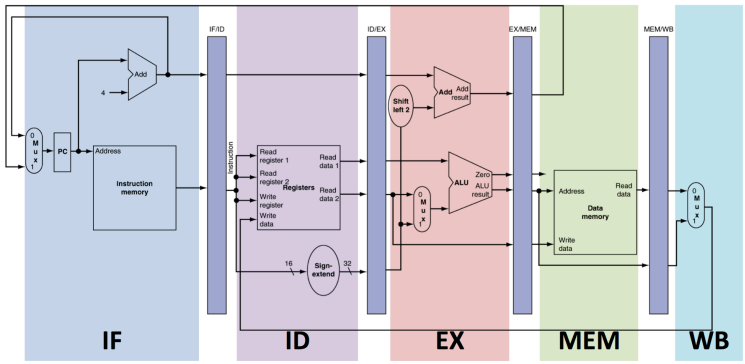
Pipelined processor finishes one instruction every 100/20+2 ns, which is every 7ns, which would be 100/7=14.3x. But it is 20% slower, because 20% of the time it doesn't finish an instruction, so the throughput is really 14.3*0.8=11.4x.

39. What is the key thing that makes pipeline processors faster?
`2015-Apr-Pipelining-1`
    a. They can execute instructions in parallel
    b. Instructions can take different numbers of cycles
    c. Instructions are split into smaller steps
    d. All of the above

Executing instructions in parallel is the key to pipeline performance. We use each part of the processor for a different instruction at the same time.

40. What value will be in R9 after this code finishes executing on the pipeline shown below?
```
R1=1, R4=1, R9=1
beq R1, R4, skip
```

```
addi R9, R0, 0
addi R9, R9, 2
addi R9, R9, 4
skip:
add R0, R0, 0
end:
```



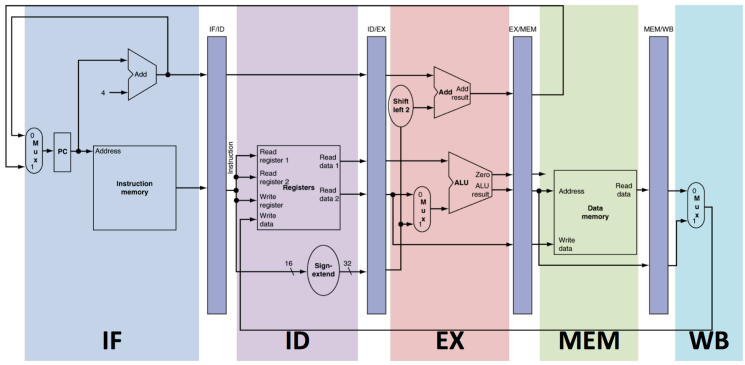
2015-Apr-Pipelining-2

a. 0
b. 1
c. 6
d. 12

The pipeline as drawn has three branch delay slots, so we will execute all three addis before we notice that we should have jumped to skip. We will get R9=1 (beq) 0, 2, 6.

41. What value will be in R9 after this code finishes executing on the pipeline shown below?

```
R1=1, R4=1, R9=1
beq R1, R4, skip
addi R9, R0, 0
skip:
addi R9, R9, 2
addi R9, R9, 4
add R0, R0, 0
end:
```



2015-Apr-Pipelining-3

a. 0
b. 1
c. 6
d. 12

The pipeline as drawn has three branch delay slots, so we will execute all three addis before we notice that we should have jumped to skip. We will then jump to skip and do the last two addis again! We will get R9=1 (beq) 0, 2, 6, 8, 12.

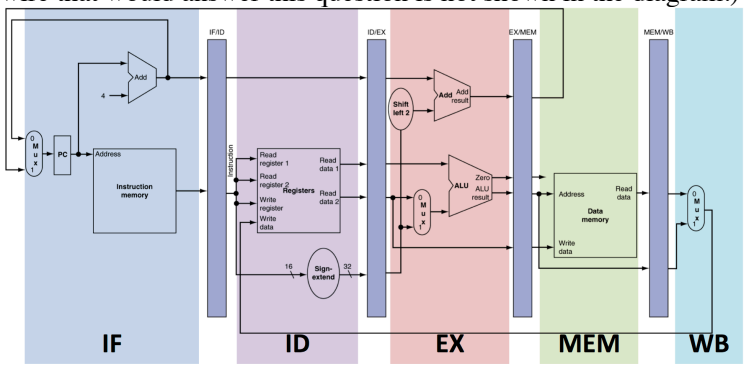
42. A processor takes 280ns for the longest instruction. The processor is pipelined with 14 (equal) stages using pipeline registers that take 10ns. What percentage of the resulting cycle time is used for computation?

2015-Apr-Pipelining-4

a. 33%
b. 66%
c. 96%
d. 100%

Each stage is 280/14=20ns plus 10ns register overhead, or 30ns. 20/30ns is used for computation, or 66%.

43. In the pipeline below, why do we need to store the ALU's Zero output in the EX/MEM stage pipeline register? (Note the wire that would answer this question is not shown in the diagram.)



2015-Apr-Pipelining-5

a. We don't
b. We need it in the MEM stage
c. We need it in the EX stage
d. <u>We need it in the IF stage</u>

The Zero value is stored so that the IF stage can use it on the cycle after the EX stage to choose between PC+4 and the branch PC.

44. Does the pipelined processor keep the ISA's promise of in-order and atomic execution?

2015-Apr-Pipelining-6

a. Atomic: yes, In-order: yes
b. Atomic: yes, In-order: no
c. Atomic: no, In-order: yes
d. <u>Atomic: no, In-order: no</u>

The pipeline executes instructions in neither an atomic nor in-order manner. The instructions now happen in 5 separate pieces (not atomically) and don't happen in-order (e.g., one instruction's EX may happen before another instruction's MEM).

45. What speedup will we get if we divide each instruction in a single-cycle processor into 8 pieces and run the clock 8 times faster?

2013-Jan-Pipelining-1

a. 1/8th the speed
b. <u>Same speed</u>
c. 8x faster
d. Depends on how many stalls we need to handle hazards

46. A single-cycle processor takes 200ns per instruction. If we pipeline it to 20 stages, and each pipeline register has 2ns delay, how much faster can it run?

2013-Jan-Pipelining-2

a. Same speed
b. 14.3 x faster
c. <u>16.7 x faster</u>
d. 20.0 x faster

47. In the previous question, how long does each instruction now take?

2013-Jan-Pipelining-3

a. 12ns
b. 14ns
c. 200ns
d. <u>240ns</u>

48. Why do we need to store Register File output 2's data in the pipeline register for the MEM stage?

2013-Jan-Pipelining-4

a. <u>Needed for the store instruction's data</u>
b. Needed for the store instruction's address
c. Needed for writeback later in the WB stage

d. We don't need to store it

49. A single-cycle processor has three types of instructions that take different amounts of time and make up different amounts of the program. (30% Type 1=600ps, 30% Type 2=700ps, and 40% Type 3=800ps.) How much faster or slower is a pipelined processor with a 200ps clock, assuming no latch overhead and no hazards?
`2013-Jan-Pipelining-5`
    a. 0.3x
    b. 1.0x
    c. 3.6x
    d. 4.0x

50. A 6-stage pipeline processor runs at 6x the clock speed of a single-cycle processor. Assume no hazards and no overhead. How much faster can the pipelined processor execute 6 instructions than the single-cycle version?
`2013-Jan-Pipelining-6`
    a. 1.0x
    b. 3.3x
    c. 6.0x
    d. 36.0x

51. Why do single-cycle processors have slower clock speeds than pipelined processors?
`2015-Aug-Pipelining-1`
    a. They are limited by the slowest instruction
    b. They have instructions that take different lengths of time
    c. They can only execute one instruction at a time
    d. All of the above

    The slowest instruction still has to finish in one cycle on a single-cycle processor, which is why the clock speed has to be slow.

52. What is needed from a pipelined processor to get better performance than a single-cycle one?
`2015-Aug-Pipelining-2`
    a. Keep the pipeline full
    b. Execute multiple instructions in parallel
    c. A higher clock speed
    d. All of the above

    The clock speed by itself is not enough to get better performance since each clock cycle does less work. The pipeline has to execute instructions in parallel, and, in particular be full of instructions to use the available parallelism.

53. How much faster does a processor run if we divide each instruction into 8 cycles, run the clock 4 times faster, and only run one instruction at a time?
`2015-Aug-Pipelining-3`
    a. 32x faster
    b. 2x faster
    c. 2x slower
    d. 32x slower

    Each instruction takes 8 cycles to finish, and each cycle is 1/4 as long, so it will take 8/4 or 2x longer to finish each instruction. We need to run them in parallel to get a pipelined benefit.

54. Why do we need to keep the RF2 data in the EX/MEM pipeline register?
`2015-Aug-Pipelining-4`
    a. For the ALU to handle R-type instructions
    b. For the memory data
    c. For the pipeline to write back to the register file
    d. We don't

    The RF2 output is the data we write to the memory in a store instruction. We need to keep it in the pipeline until we get to the MEM stage so we have the data to write to the memory.

55. A single-cycle processor takes 80ns per instruction. It is perfectly pipelined into 10 stages with pipeline registers that take 2ns each and can be kept 50% full. What change in instruction throughput do you expect compared to the single-cycle processor?

2015-Aug-Pipelining-5

a. 10x
b. 5x
c. 4x
d. 2x

Each stage takes 80ns/10=8ns to do work and 2ns for the pipeline register, so it will be 10ns per stage. If the pipeline was 100% full, our throughput speedup would be 80ns/10ns=8x. If the pipeline is 50% full, that means we will finish an instruction 50% of the time, so our speedup will be 50% of that, or 4x higher throughput.

56. A single-cycle processor takes 80ns per instruction. It is perfectly pipelined into 10 stages with pipeline registers that take 2ns each and can be kept 50% full. What change in instruction latency do you expect compared to the single-cycle processor?

2015-Aug-Pipelining-6

a. 1.0x
b. 1.25x
c. 0.75x
d. 0.5x

The latency was 80ns for an instruction to finish before. Now each instruction will finish in 10 stages, and each stage takes (80/10+2)=10ns each, or 100ns. So the latency is now 100ns/80ns = 1.25x. The % full doesn't affect how long each instruction takes to finish once it gets into the pipeline.

57. What is needed to get better performance from a pipelined processor?

2014-Sep-Pipelining-1

a. A higher clock speed
b. Executing multiple instructions in parallel
c. Keeping the pipeline full
d. All of the above

The clock speed by itself is not enough to get better performance since each cycle does less work. The pipeline has to execute instructions in parallel, and, in particular, be full to get the benefits.

58. What limits the clock speed for a non-pipelined processor?

2014-Sep-Pipelining-2

a. The fastest instruction path
b. The average instruction path
c. The slowest instruction path
d. The slowest logic unit

The clock speed cannot be any faster than the logic path used by the slowest instruction. This may include multiple logic units.

59. A processor is pipelined 10 times and runs at 10x the clock speed with no overhead, but only manages to keep the pipeline half full. What is the speedup?

2014-Sep-Pipelining-3

a. 5x
b. 10x
c. 50x
d. 100x

If the pipeline was full we would finish instructions 10x faster, but since we only finish instructions on every other cycle, it will only be 5x faster.

60. A single-cycle processor takes 100ns per instruction and can be perfectly divide up into pipeline stages and the pipeline can be kept completely full. However, each pipeline register has an overhead of 1ns. What degree of pipelining gives the best instruction throughput?

2014-Sep-Pipelining-4

a. 1 stage (1 pipeline register at the end)
b. 10 stages
c. 100 stages
d. 1000 stages

1000 stages will mean that you spend 100/1000=0.1ns computing per stage plus 1ns register overhead. This will give a cycle time of 1.1ns, which means you get an output every 1.1ns. 10 stages will mean you spend 100/10=10ns computing plus 1ns register overhead. This will give you a cycle time of 11ns, which means you get an output every 11ns. So the more stages the better for throughput as long as the pipeline is full.

61. A single-cycle processor takes 100ns per instruction and can be perfectly divide up into pipeline stages and the pipeline can be kept completely full. However, each pipeline register has an overhead of 1ns. What degree of pipelining gives the best instruction latency?
`2014-Sep-Pipelining-5`
    a. 1 stage (1 pipeline register at the end)
    b. 10 stages
    c. 100 stages
    d. 1000 stages

1000 stages will mean that you spend 100/1000=0.1ns computing per stage plus 1ns register overhead. This will give a cycle time of 1.1ns, which means an instruction will finish in 1000*1.1 = 1100ns. 10 stages will mean you spend 100/10=10ns computing plus 1ns register overhead. This will give you a cycle time of 11ns, which means you will finish in 10*11ns = 110ns. The lowest latency is with the minimum amount of pipeline overhead, or 1 stage=101ns.

62. How many stages do processors that run at multiple GHz have these days?
`2014-Sep-Pipelining-6`
    a. 1-10
    b. 11-25
    c. 26-50
    d. 50+

Modern processors balance the clock speed and keeping the pipeline full for best performance. This results in a sweet spot at around 20 pipeline stages.

63. What determines the clock speed of a single-cycle processor?
`2016-Oct-Pipelining-1`
    a. The shortest latency instruction
    b. The longest latency instruction
    c. The number of cycles the longest instruction takes
    d. The time to decode the longest instruction

All instructions have to be completed in a single cycle, so the slowest one determines that time.

64. Does the pipelined processor execute instructions in an atomic manner?
`2016-Oct-Pipelining-2`
    a. Yes, because it is in the ISA's contract
    b. Yes, because all instructions are always executed to completion
    c. No, the execution happens in 5 different pieces
    d. It depends on the instruction

Because instructions are executed in pieces they are not completed atomically.

65. If we split the instructions into 100 smaller chunks we could run at 100x the clock frequency. Why do we not have 100-stage pipelines in regular processors?
`2016-Oct-Pipelining-3`
    a. Hard to break instructions down to really small pieces
    b. Pipeline register overhead will kill us
    c. Hard to keep long pipelines full
    d. All of the above

Making a perfect pipeline is difficult, and keeping it full if it is too long is extremely hard. We always have pipeline register overhead and mispredicted branches that waste cycles.

66. A processor takes 120ns for the longest instruction. The processor is pipelined with 6 (equal) stages using pipeline registers that take 20ns. What percentage of the resulting cycle time is used for computation??
`2016-Oct-Pipelining-4`
    a. 22%
    b. 30%

c. 50%
d. 67%

Each stage is 120/6=20 ns + 20 ns overhead = 40 ns. 20/40 ns is used for computation, or 50%.

67. What do store word instructions do during the EX stage of the MIPS pipeline?
`2016-Oct-Pipelining-5`
    a. Nothing
    b. Add
    c. Forward pipeline registers
    d. Set the MemWrite control

The store word instruction does an add to calculate the address.

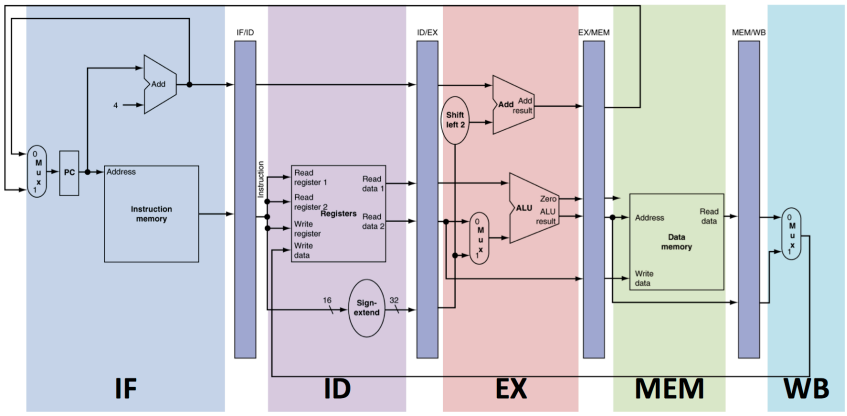
68. How does pipelining improve performance?
`2016-Oct-Pipelining-6`
    a. It divides instructions into smaller steps
    b. It allows the processor to work on multiple instructions at the same time
    c. It allows shorter instructions to finish sooner
    d. It reduces the instruction latency

Dividing up instructions makes them take longer. Shorter instructions still need to go through all the stages. Instruction latency goes up because of the overhead of pipeline registers.

69. What is required to make pipelines run at their full potential?
`2016-Jan-Pipelining-1`
    a. Keeping them full
    b. Dividing up instructions equally across stages
    c. Executing instructions in parallel
    d. All of the above

All of the above are required to make the pipeline run at its full potential: if you can't keep it full you waste cycles; if instructions aren't divided up equally you can't run the clock faster; and if you can't execute in parallel there is no point to a pipeline.

70. What data is not stored in the EX/MEM pipeline register?



`2016-Jan-Pipelining-2`
    a. Register File 2 output
    b. ALU result
    c. Branch address
    d. Sign-extended immediate

The sign-extended immediate is consumed (used) in the EX stage so it is not needed later and is therefore not stored.

71. How much faster (instruction throughput) does a processor run if we just divide each instruction into 7 stages?
`2016-Jan-Pipelining-3`
    a. 1/7=0.14x
    b. 1x
    c. 7x

d. 49x

If you divide each instruction into 7 parts it will take 7 times as long to complete each instruction, so it will run 1/7th as fast. You need to execute the instructions in parallel and also increase the clock speed by 7x to get the 7x speedup.

72. How many branch delay slots are needed in the pipeline shown above?
`2016-Jan-Pipelining-4`
    a. 0
    b. 1
    c. 2
    d. 3

The result of the branch computation doesn't get returned to the IF stage until MEM, which is 3 cycles behind, so we need 3 branch delay slots.

73. A single-cycle processor takes 100ns for each instruction and a pipeline register takes 2ns. What is the cycle time for a pipelined version of the processor if you can divide the instructions up into 100 equal stages?
`2016-Jan-Pipelining-5`
    a. 1.02ns
    b. 1ns
    c. 2ns
    d. 3ns

100ns divided into 100 stages is 1ns for the instruction, but you also have to include the 2ns for the pipeline register.

74. What does the pipeline do that causes problems with the ISA's promise of atomic execution?
`2016-Jan-Pipelining-6`
    a. Nothing: there is no problem
    b. It executes parts of instructions across multiple stages
    c. It executes multiple instructions at the same time
    d. B and C

Because instructions are split across multiple stages and multiple instructions are executed at the same time, we can end up with instructions requesting data that has not been written back.

75. Why did pipelining the processor help if all instructions take the same 5 pipeline stages?
`2013-Aug-Pipelining-1`
    a. We can execute the instructions in parallel
    b. We can execute each instruction faster
    c. We can move data through the processor faster
    d. It didn't: pipelining only helps when the instructions take different amounts of time

Pipelining allows us to execute instructions in parallel. (E.g., one instruction can be doing ID while the second one is doing IF.) Each individual instruction may not be faster, but we can do more at once by doing them in parallel.

76. What is the speedup for a processor when we add 10 ideal pipeline stages, but can only keep them half full with instructions?
`2013-Aug-Pipelining-2`
    a. 10x
    b. 5x
    c. 2x
    d. Need more information

The ideal speedup for 10 pipeline stages is 10x, but if we only keep them half full, then that means we only finish an instruction on every other cycle, so we have half the benefit, or a speedup of 5x.

77. What is the change in throughput for a processor that takes 100ns per instruction unpiplined, when it is pipelined with 100 stages, where each stage needs a 2ns pipeline register? (Assume we can keep the pipeline completely full.)
`2013-Aug-Pipelining-3`
    a. 1/3 (lower throughput)
    b. 3x

c. 33x
d. 100x

We will get one instruction out every 100ns/100+2ns, or 3ns. So this is 33x higher throughput than the 1 every 100ns we started with.

78. What is the change in latency to complete each individual instruction for a processor that takes 100ns per instruction unpiplined, when it is pipelined with 100 stages, where each stage needs a 2ns pipeline register? (Assume we can keep the pipeline completely full.)
2013-Aug-Pipelining-4
    a. 1/3 (lower latency)
    b. 3x
    c. 33x
    d. 100x

Each instruction needs 100 stages, where each stage takes 100ns/100 logic and 2ns pipline, or 3ns per stage times 100 stages is 300ns. Previously each instruction took 100ns, so it is 3x larger.

79. What happens in the MEM stage for an addi instruction?
2013-Aug-Pipelining-5
    a. Nothing
    b. Load the value
    c. Calculate the ALU result
    d. Write back the ALU result

Nothing. The addi instruction does not access the data memory, so it does nothing in the MEM stage.

80. What is the most important thing for getting performance from a pipelined design?
2013-Aug-Pipelining-6
    a. Have as few pipeline registers as possible
    b. Make each stage fast
    c. Keep the pipeline full
    d. All of the above

If you can't keep the pipeline full then the others don't matter.

81. How much faster does a processor run if we divide each instruction into 7 parts and run the clock 7x faster?
2013-Apr-Pipelining-1
    a. 1x
    b. 7x
    c. 49x
    d. Depends on the branch predictor

Each instruction is 7x faster, but we have 7x as many = 7/7=1x faster. We need to run the instructions in parallel to get a speedup!

82. What is the key to getting performance out of a pipeline?
2013-Apr-Pipelining-2
    a. Predicting branches
    b. Avoiding branch delay slots
    c. Keeping the pipeline full
    d. Re-ordering instructions

A, c, and d are all ways to keep the pipeline full.

83. A processor takes 180ns for the longest instruction. The processor is pipelined with 6 (equal) stages using pipeline registers that take 10ns. What percentage of the resulting cycle time is used for computation?
2013-Apr-Pipelining-3
    a. 25%
    b. 75%
    c. 85%
    d. 95%

Each stage is 180/6=30ns plus 10ns register overhead, or 40ns. 30/40ns is used for computation, or 75%.

84. A processor takes 100ns for the longest instruction and pipeline registers that take 2ns. What is the lowest cycle time you could get for the processor if the processor can be divided up into an infinite number of equal stages?
`2013-Apr-Pipelining-4`
    a. ~0ns
    b. ~2ns
    c. ~50ns
    d. ~100ns

As you get an infinite number of stages, your instruction time goes down to 100ns/infinity=0, and you're just left with the pipeline registers, which are 2ns. (This is just Amdahl's law.)

85. A processor takes 180ns for the longest instruction. The processor is pipelined with 6 (equal) stages using pipeline registers that take 10ns. How much more or less time does each instruction take compared to the un-pipelined processor? (You can assume there are N pipeline registers for N stages.)
`2013-Apr-Pipelining-5`
    a. 140ns shorter
    b. same
    c. 10ns longer
    d. 60ns longer

We have 6 stages that each take 180/6+10ns, so the total is 40ns*6 = 240ns. This is 60ns longer than the 180ns it took before.

86. What do load word instructions do during the EX stage of the MIPS pipeline?
`2013-Apr-Pipelining-6`
    a. Nothing
    b. Add
    c. Forward pipeline registers
    d. Set the MemWrite control

The load word instruction does an add to calculate the address.

7. Hazards

1. When can we forward data in a pipeline?
`2013-Dec-Hazards-1`
    a. When the data is in the wrong place
    b. When we need the data from an instruction before us
    c. When multiple instructions need the same hardware at once
    d. All of the above

The data has to already be computed, but not in the register file for forwarding to help. We can't forward from an instruction before us if it hasn't computed the data. We can't forward around structural hazards (needing the same hardware for multiple instructions).

2. Hazard detection logic for a pipeline stage needs to look at what?
`2013-Dec-Hazards-2`
    a. The current instruction and earlier instructions in the pipeline
    b. The current instruction and later instructions in the pipeline
    c. All instructions in the pipeline
    d. Just the current instruction

Hazard detection needs to look and see if the data for the instruction is coming from an earlier instruction that has not completed yet. This means it needs to look at the current instruction and earlier instructions in the pipeline.

3. What forwarding is needed for this code?
```
add R3, R2, R0
sub R2, R3, R0
and R3, R3, R2
```
`2013-Dec-Hazards-3`

a. MEM to EX
b. WB to EX
c. MEM to EX and WB to EX
d. None

sub requires data from the instruction before it (MEM to EX) and and requires data from both the instruction before it (MEM to EX) and the instruction two before it (WB to EX).

4. lw R3, 0(R2)
   add R4, R3, R2
   Why can't we forward to avoid the load delay slot between a lw and add instruction?
   2013-Dec-Hazards-4
   a. We can if we add another path
   b. The data is not available when the add needs it
   c. It's a structural hazard so we need to add more hardware
   d. We would break the serial and atomic ISA promise

   The lw is accessing the data in the same cycle (but in the MEM stage) as the add needs it (in the EX stage). Since the data is not actually available there is nothing we can do but wait.

5. Which of the following hazards is not a structural hazard?
   2013-Dec-Hazards-5
   a. Calculating the branch condition and address at the same time
   b. Reading and writing the register file at the same time
   c. Reading the instructions and data at the same time
   d. Writing the data read by the previous instruction

   Writing the data read by the previous instruction is a data hazard and just requires a new forwarding path. The others all require extra hardware to do multiple things at the same time

6. What is the performance improvement (if any) of being clever (if you can) and filling the nop branch delay slot in this code?
   ```
   add R4, R5, R4
   add R3, R6, R5
   beq R3, R0, label
   nop
   sub R3, R3, R4
   ```
   2013-Dec-Hazards-6
   a. No improvement
   b. Can't reorder
   c. 20% faster
   d. 40% faster

   We can move the first add into the nop slot which reduces the total instructions from 5 to 4, which is a 20% improvement.

7. When can we forward data in a pipeline?
   2017-Jan-Hazards-1
   a. Whenever we need data from an instruction before us
   b. When multiple instructions need the same hardware at the same time
   c. When the data is in another stage
   d. All of the above

   The data has to already be computed, but not in the register file, for forwarding to help. We can't forward from an instruction before us if it hasn't computed the data. We can't forward around structural hazard (needing the same hardware for multiple instructions).

8. What would be a structural hazard?
   2017-Jan-Hazards-2
   a. Needing a result that is in another part of the processor
   b. Needing a result that can't be computed because the hardware is busy
   c. Computing a result for an instruction that should not have been executed
   d. Any of the above

A result that is in another place is a data hazard. Computing a result that shouldn't be computed is a control hazard. Needing a result that can't be computed because the hardware is busy is a structural hazard. This can be solved by adding more hardware.

9. What is needed to resolve the following hazard?
   and R1,R2,R1
   ld R2,0(R3)
   add R2,R4,R1
   2017-Jan-Hazards-3
   a. Forward from MEM to EX
   b. Forward from WB to EX
   c. A and B are both required
   d. Nothing, there is no hazard

   The third instruction needs R1 in the EX stage, but the results from the first instruction is still in in the WB stage, so we need to forward from WB to EX..

10. The bcopy routine does a bit copy from one memory location to another. The code consists of the following loop. What forwarding paths are needed in the standard 5-stage MIPS pipeline to avoid data hazards in this code?
    loop:
    addi R4, R4, 4 # increment source address
    addi R5, R5, 4 # increment destination address
    lw R6, 0(R4) # load source data
    sw R6, 0(R5) # store data to destination
    bne R4, R7, loop # check if we are done with the copy
    2017-Jan-Hazards-4
    a. WB-EX, MEM-EX
    b. WB-EX, WB-MEM
    c. MEM-EX, WB-MEM
    d. WB-EX, MEM-EX, WB-MEM

    For both addi->lw we need to forward from the WB to the EX stage. For lw->sw we need to forward from the WB-MEM stage.

11. An 8-stage pipeline processor requires NOPs to avoid hazards. The compiler is aware of this and inserts appropriate NOP instructions into the code. What would happen if the processed code now is run on a single-cycle processor?
    2017-Jan-Hazards-5
    a. The program would execute correctly
    b. Conditional branches would not work correctly, due to the zero output from the ALU
    c. The code would not work correctly because the NOPs are in the wrong place
    d. The code would not work correctly because there are too many NOPs

    Everything would execute correctly since a single-cycle processor does not have any hazards, however the NOPs would make the program run slower than necessary.

12. In a certain program run on a 5-stage pipeline processor with double-pumped register file but no forwarding, 50% of all instructions are loads and 10% of the instructions directly following the loads depend on it. What is the slowdown of the program due to these dependencies?
    2017-Jan-Hazards-6
    a. 1.0x (no slowdown)
    b. 1.05x
    c. 1.1x
    d. 1.2x

    You will need 2 bubbles to handle the data dependency, so the dependent instructions will have a throughput of one-third IPC, or 3 cycles per instruction. 0.5*0.10*3+(1?0.5*0.10)*1=1.1.

13. Forwarding helps when?
    2014-Apr-Hazards-1
    a. If the answer is somewhere else in the processor
    b. If the answer hasn't been computed yet
    c. If multiple parts of the processor need the answer at the same time
    d. All of the above

Forwarding can only solve the problem that the data isn't where we need it.

14. Hazards are a result of what?
    2014-Apr-Hazards-2
    a. Needing to keep the semantics of atomic and sequential execution
    b. Executing instructions in parallel
    c. Executing instructions in pieces
    d. All of the above

    If we didn't have to keep the atomic and sequential semantics of the ISA then we wouldn't have any hazards. (The programmer would just have to figure it out.) But since we do need to keep those semantics, the parallel execution of pieces of instructions causes hazards.

15. What forwarding paths are needed to handle this code correctly?
    ```
    lw R12, 0(R3)
    add R2, R12, R3
    add R3, R14, R8
    ```
    2014-Apr-Hazards-3
    a. None
    b. MEM-->EX
    c. WB-->EX
    d. Can't forward here

    You can't forward here because the result of the lw is available in the MEM stage in the same cycle that the add needs it in the EX stage. That is, the result is not available when you need it so you can't forward.

16. What is the CPI (cycles per instruction) for a program where 30% of all instructions are loads and 1/3 of the instructions following the load depend on it?
    2014-Apr-Hazards-4
    a. 1.01
    b. 1.1
    c. 1.3
    d. 1.33

    We need to insert a NOP 1/3 of 30% of the time, or 10%. When we do that our CPI goes to 2. So we have a CPI of 1*90%+ a CPI of 2*10% = a CPI of 1.1 or a 10% slowdown.

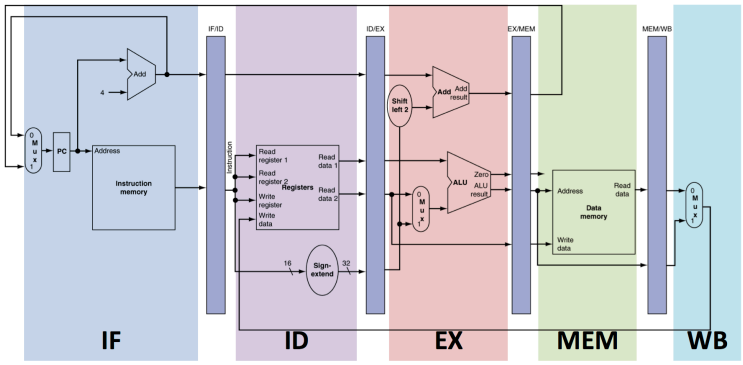
17. Which of these does not solve a structural hazard?
    2014-Apr-Hazards-5
    a. Data forwarding
    b. Separate comparator for branch detection
    c. Having separate instruction and data memories
    d. Double-pumping the register file

    Data forwarding addressed a data hazard. All the others either added hardware or modified hardware to allow multiple instructions to execute at the same time.

18. Was the decision to have a branch delay slot in the MIPS ISA a good one in retrospect?
    2014-Apr-Hazards-6
    a. Yes, it allowed the compilers to produce faster code
    b. Yes, but it was hard to use and required hand-written code
    c. No, because it was too hard to use so no one every filled it
    d. No, because newer processors had different pipelines but had to support the ISA

    A single branch delay slot wasn't too hard to use. The problem was that future pipelines had different numbers and had to have special logic to pretend they had just one slot.

19. What is needed to resolve the following hazard?
    ```
    and R1,R2,R1
    ld R2,0(R3)
    add R2,R4,R1
    ```



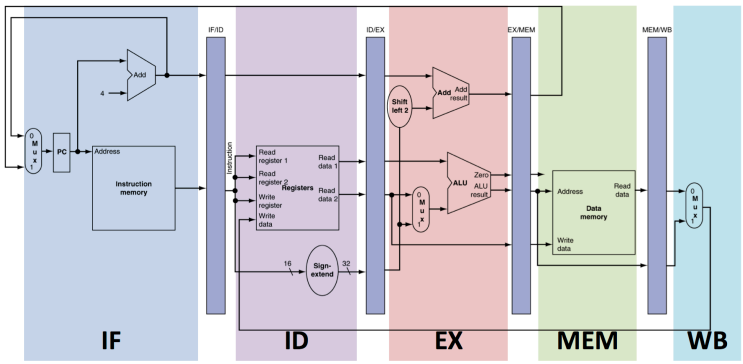
2015-Oct-Hazards-1

a. Forward from MEM to EX
b. Forward from WB to EX
c. A and B are both required
d. Nothing, there is no hazard

The third instruction needs R1 in the EX stage, but the results from the first instruction is still in in the WB stage, so we need to forward from WB to EX..

20. A 5-stage MIPS pipeline has a double-pumped register file and forwarding from WB to EX but not from MEM to EX. How many NOPs will you need to add to make this code work?

```
add R10, R12, R13
sub R12, R13, R11
add R13, R12, R11
```



2015-Oct-Hazards-2

a. 3
b. 2
c. 1
d. 0

You need a single NOP to use the forwarding for the R12 dependency.

21. An 8-stage pipeline processor requires NOPs to avoid hazards. The compiler is aware of this and inserts appropriate NOP instructions into the code. What would happen if the processed code now is run on a single-cycle processor?

2015-Oct-Hazards-3

a. The program would execute correctly
b. Conditional branches would not work correctly, due to the zero output from the ALU
c. The code would not work correctly because the NOPs are in the wrong place
d. The code would not work correctly because there are too many NOPs

Everything would execute correctly since a single-cycle processor does not have any hazards, however the NOPs would make the program run slower than necessary.

22. What kind of hazard would an add instruction experience a in pipeline that executes two instructions at a time (dual-issue) that it does not experience in our standard (single-issue) pipeline?

2015-Oct-Hazards-4

a. Structural hazard
b. Data hazard
c. Control hazard
d. None

If two instructions can be executed at the same time you may need two adders to do them both. This is a structural hazard. The regular pipeline already had data hazards for the add instruction, and adds do not have control hazards.

23. Which of the following is true about hazards?
`2015-Oct-Hazards-5`
    a. All hazards can be solved with bubbles
    b. The use of bubbles breaks the ISA promise of "Atomic execution" of instruction
    c. Using the branch delay slot ensures the promise of sequential execution is kept
    d. There are still control hazards in single-cycle processor

Delaying instructions long enough that the processor is effectively not pipelined will always solve a hazard, but will hurt performance.

24. In a certain program run on a 5-stage pipeline processor with double-pumped register file but no forwarding, 50% of all instructions are loads and 10% of the instructions directly following the loads depend on it. What is the slowdown of the program due to the stalls needed to handle these dependencies?
`2015-Oct-Hazards-6`
    a. 1.0x (no slowdown)
    b. 1.05x
    c. 1.1x
    d. 1.2x

You will need 2 bubbles to handle the data dependency, so the dependent instructions will have a throughput of one-third IPC, or 3 cycles per instruction. 0.5*0.10*3+(1-0.5*0.10)*1=1.1.

25. When is forwarding required in a non-pipelined processor?
`2018-Oct-Hazards-1`
    a. Never, forwarding is only required in pipelined processors
    b. If the ISA promises atomic and sequential execution
    c. When an instruction depends on the one immediately before it
    d. After a branch instruction

The data has to already be computed, but not in the register file, for forwarding to help. We can't forward from an instruction before us if it hasn't computed the data. We can't forward around structural hazard (needing the same hardware for multiple instructions).

26. Which of the following does not create structural hazards?
`2018-Oct-Hazards-2`
    a. Comparing branch equality in the ALU
    b. Reading and writing the ALU on each cycle
    c. Loading data and instructions from memory
    d. Needing data from a later pipeline stage

Needing data from a later pipeline stage is not a problem with not having enough hardware to do the instruction (structural hazard), but rather a problem with not having the data in the right place (forwarding).

27. Why are hazards a problem in our MIPS pipeline?
`2016-Aug-Hazards-1`
    a. Multiple instructions execute in parallel
    b. The programs are written assuming the ISA's promise of sequential and atomic execution
    c. Instructions are executed in multiple separate pieces
    d. All of the above

If the ISA did not promise sequential and atomic execution (and we did not execute things in parallel and in separate pieces) then the programmer would have a much harder time, but we wouldn't have any hazards the processor has to handle. But to keep the ISA's promise we do have to handle hazards.

28. When does replicating logic help with hazards?
`2016-Aug-Hazards-2`
    a. If the answer is somewhere else in the processor
    b. If the answer hasn't been computed yet
    c. If multiple parts of the processor need to do the same operation at the same time

d. All of the above

Replicated logic allows multiple parts of the processor to do the same thing at the same time. For example, read the register file and write, or add for a branch and for the ALU.

29. What is needed to resolve any hazards in this example?

```
add R14, R1, R13
lw R14, 4(R2)
sw R14, 4(R13)
addi R2, R13, R1
```

2016-Aug-Hazards-3

a. Forwarding from MEM to EX
b. Forwarding from WB to MEM
c. Can't resolve; need to stall
d. There are no hazards to resolve

The only data dependency between these instructions is R14 between the lw and the sw. To handle that hazard we need to forward from WB to MEM.

30. What is needed to resolve any hazards in this example?

```
addi R14, R14, 8
lw R1, 4(R13)
add R1, R1, R13
sw R14, 4(R13)
```

2016-Aug-Hazards-4

a. Forwarding from MEM to EX
b. Forwarding from WB to MEM
c. Can't resolve; need to stall
d. There are no hazards to resolve

The only dependencies are between the first addi and the last sw, but this is far enough to not require any forwarding, and between the lw and the add. However, the lw doesn't get its data back from the memory until after the MEM stage so there is no way to forward it to an instruction in the previous stage, so we cannot solve this with forwarding.

31. A processor pipeline is ideally pipelined to 20 stages, but has 5 branch delay slots as a result. If the program has 10% branches and the compiler can only fill 2/5 branch delay slots, what is the expected performance?

2016-Aug-Hazards-5

a. 10x
b. 19.7x
c. 20x
d. 23x

We would expect 20x performance, but 10% of the time we have a branch, and each time we have a branch we leave 3/20 pipeline stages empty. So on average we have 10%*3/20=0.3/20 stages empty, or 19.7/20 full. So the performance is (19.7/20)*20 = 19.7x .

32. Why was making the branch delay slot a part of the MIPS ISA a really bad decision?

2016-Aug-Hazards-6

a. It forced future processors to pretend to have one delay slot no matter what they really had
b. It forced compiler writers to figure out how to always fill the branch delay slot for every branch
c. It made MIPS code a lot larger if it used any branches
d. All of the above

Compilers do not have to fill the slot, but they give up some performance if they don't. The code is only larger if you can't fill the slot. Neither of these are really bad. The biggest problem (by far) was that it forced all future MIPS pipelines to act as if they have exactly one delay slot no matter how many they really had.

33. What is needed to resolve the following hazard?

```
xor R12, R13, R12
ld R2, R3
add R13, R4, R12
```

2014-Dec-Hazards-1

a. Double-pumping the register file
b. Forwarding from WB to EX
c. Forwarding from MEM to EX
d. There is no hazard to resolve, there are enough instructions between the dependencies.

The third instruction needs R12 in the EX stage, but the results from the previous instruction are only in the MEM stage, so we need to forward from MEM to EX.

34. Which of the following statements is true about hazards?
2014-Dec-Hazards-2
    a. The most efficient way to solve hazards is stalling the pipeline
    b. A compiler can solve all hazards without needing to insert nops
    c. Performance is hurt a lot if hazards are solved by stalling the pipeline
    d. All hazards can be solved forwarding data between stages.

A compiler cannot solve all hazards by inserting nops, since sometimes it cannot find enough independent instructions to put between dependent ones. Not all hazards can be solved by forwarding data. For example, if the value hasn't been computed yet. Stalling the pipeline is a bad solution, since it hurts performance a lot.

35. The instructions in the branch and load delay slots after the branch or load don't see the effect of the branch or load before they execute. Does this follow the ISA's promise of atomic and sequential execution?
2014-Dec-Hazards-3
    a. Yes
    b. Atomic yes, sequential no
    c. Atomic no, sequential yes
    d. Neither atomic nor sequential

These instructions act as if they happen before the branch or load, so this is not sequential execution. You do not see any impact of the branch/load executing at the same time, so they are still effectively atomic.

36. A 5-stage MIPS pipeline has a double-pumped register file and forwarding from MEM to EX but not from WB to EX. How many NOPs will you need to add to make this code work?
```
add R1, R2, R3
add R2, R3, R1
add R3, R1, R2
```
2014-Dec-Hazards-4
    a. 0
    b. 1
    c. 2
    d. 3

With no forwarding from WB to EX, we cannot use results that are two instructions behind us without inserting a NOP to wait for it to be in the register file. So the R1 from the first instruction can be used in the second instruction and the R2 from the second instruction can be used in the third instruction, but to use the R1 from the first instruction in the third instruction we need to insert a NOP between the 2nd and 3rd instruction, which means that we can no longer forward the R2, and we need another NOP.

37. What is the logic to check if we need to forward from MEM to EX? (Rs, Rt are the sources and Rd is the destination.)
2014-Dec-Hazards-5
    a. (EX.Rs == MEM.Rd) OR (EX.Rt == MEM.Rd)
    b. (EX.Rd == MEM.Rs) OR (EX.Rd == MEM.Rs)
    c. (EX.Rs == MEM.Rs) AND (EX.Rt == MEM.Rt)
    d. (EX.Rs == MEM.Rt) OR (EX.Rs == MEM.Rs) OR

(EX.Rt == MEM.Rt) OR (EX.Rt == MEM.Rs)
We need to see if the destination of the instruction in the MEM stage is the same as either of the sources of the instruction in the EX stage..

38. What forwarding paths do we need to execute this code with no NOPs on a processor with a double-pumped register file?
```
add R1, R2, R3
lw R2, 0(R1)
sw R2, 0(R3)
sub R2, R1, R3
```

2014-Dec-Hazards-6

a. MEM-EX, WB-EX
b. MEM-EX, WB-EX, WB-MEM
c. MEM-EX, WB-MEM
d. WB-EX, WB-MEM

From add to lw we need MEM to EX, from lw to sw we need WB to MEM, and from add to sub we don't need anything.

39. Double-pumping the register file fixes which of the following problems?

2015-Apr-Hazards-1

a. Needing to read the register file early for branches
b. Needing to get data to the register file early for all dependent instructions
c. Needing to read and write the register file at the same time
d. All of the above

Double pumping just allowed us to read and write in the same cycle from different pipeline stages. It did not fix all of the dependent instructions.

40. What is needed to resolve the following hazard?

```
add R14, R1, R13
lw R1, 4(R14)
add R14, R15, R15
addi R14, R12, 0
```

2015-Apr-Hazards-2

a. Forwarding from MEM to EX
b. Forwarding from WB to MEM
c. Can't resolve, need to stall
d. There is no hazard to resolve

The second instruction needs R14 in the EX stage to calculate the address and the result is in the MEM stage, so we need to forward from the MEM stage to the EX stage.

41. What is needed to resolve the following hazard in the standard 5-stage pipeline?

```
lw R1, 4(R14)
add R14, R1, R13
add R14, R15, R15
addi R14, R12, 0
```

2015-Apr-Hazards-3

a. Forwarding from MEM to EX
b. Forwarding from WB to MEM
c. Can't resolve, need to stall
d. There is no hazard to resolve

The second instruction needs R1 in the EX stage, but the results of R1 from the previous instruction are only available after the MEM stage. So we can't forward them (they don't exist yet) and have to stall.

42. Which of the following options contain only structural hazards?

2015-Apr-Hazards-4

a. Calculating the branch condition and address at the same time; reading and writing the RF at the same time; writing the data read by the previous instruction.
b. Reading the instructions and data at the same time; writing the data read by the previous instruction.
c. Calculating the branch condition and address at the same time; reading the instructions and data at the same time; writing the data read by the previous instruction.
d. Calculating the branch condition and address at the same time; reading and writing the RF at the same time; reading instructions and data at the same time.

Writing the data read by the previous instruction is a data hazard and just requires a new forwarding path. The others all require extra hardware to do multiple things at the same time, which is a structural hazard.

43. A processor is ideally pipelined to 10 stages, but due to hazards, 10% of the instructions are NOPs. What is the speedup over the original processor?

2015-Apr-Hazards-5

a. 10%
b. 90%
c. 9x
d. 10x

If the pipeline was full, an ideally pipelined processor with 10 stages will be 10x faster. But we waste 10% of our instructions, so we only realize 90% of the speedup, or 90%*10=9x.

44. Which of these does not address a structural hazard?
2015-Apr-Hazards-6
    a. Having independent data and instruction memories
    b. Reading and writing the register file in the same cycle
    c. Extra comparator for branch decisions
    d. Forwarding from a later stage

Data forwarding addresses a data hazard (the data is somewhere else). All the others add hardware or modify hardware to allow multiple instructions to execute at the same time.

45. Which one of these is not a hazard
2013-Jan-Hazards-1
    a. Data hazard
    b. Control hazard
    c. Pipeline Register hazard
    d. Structural hazard

46. Double-pumping the register file fixed what kind of hazard?
2013-Jan-Hazards-2
    a. Data hazard
    b. Control hazard
    c. Pipeline Register hazard
    d. Structural hazard

47. How many hazards does the code below have for the standard 5-stage MIPS pipeline without any forwarding logic?
```
add r1, r2, r3
add r4, r2, r2
ld r5, 0(r4)
add r4, r5, r4
```
2013-Jan-Hazards-3
    a. 0
    b. 1
    c. 2
    d. 3

48. How many bubbles are needed in the pipeline to correctly execute the following code assuming no forwarding?
```
add r1, r2, r5
sub r2, r2, r3
add r3, r5, r5
ld r4, 0(r1)
beq r1, r2, done
add r1, r2, r3
sub r2, r2, r4
```
2013-Jan-Hazards-4
    a. 0
    b. 1
    c. 2
    d. 3

49. What is the CPI if 20% of the instructions generate results used by the next instruction and there is no forwarding? (Assume there are no other hazards in the pipeline.)
2013-Jan-Hazards-5
    a. 0.6
    b. 1.0
    c. 1.2

d. 1.4

50. Why couldn't we move the branch computation logic even further forward in the pipeline?
2013-Jan-Hazards-6
    a. We need to read the register file
    b. We can't put four ALUs in one stage
    c. We need to put the instruction in the pipeline register
    d. We can't update the PC while reading the instruction memory

51. When can we forward data in a pipeline?
2015-Aug-Hazards-1
    a. Whenever we need data from an instruction before us
    b. When multiple instructions need the same hardware at the same time
    c. When the data is in another stage
    d. All of the above

    The data has to already be computed, but not in the register file, for forwarding to help. We can't forward from an instruction before us if it hasn't computed the data. We can't forward around structural hazard (needing the same hardware for multiple instructions).

52. Why are hazards a problem?
2015-Aug-Hazards-2
    a. Because we only have one register file
    b. Because the ISA promises sequential execution
    c. Because writeback happens at the end of the pipeline
    d. Because the ISA promises atomic execution

    The ISA promises that instructions will execute atomically (that is, each one finishes before the next) but pipelining starts the next one before the previous one finishes.

53. Why can we not forward between these instructions to execute them back-to-back?
    lw R3, 12(R1)
    add R2, R1, R3
2015-Aug-Hazards-3
    a. It's a structural hazard so we need to add more hardware
    b. We can forward if we add another forwarding path
    c. We would break the serial and atomic ISA promise
    d. The data is not available when the add would need it

    The lw is reading the memory data in the same cycle (but in the MEM stage) as the add needs it (in the EX stage). Sicne the data is not actually available when the add needs it there is nothing we can do but wait.

54. How many register values need to be forwarded in the following code?
    xor R3, R4, R7
    add R6, R4, R5
    sub R2, R3, R4
    beq R2, R3, done
2015-Aug-Hazards-4
    a. 0
    b. 1
    c. 2
    d. 3

    R3 (xor->sub), R2 (sub->beq). We do not need to forward R3 to beq because it is far enough away that it will be in the register file. R6 is not used by any of the instructions so it does not need to be forwarded.

55. The standard MIPS pipeline with no forwarding has 50% of the instructions use data from the previous instruction. On average, the pipeline finishes one instruction every how many cycles?
2015-Aug-Hazards-5
    a. 1.0
    b. 1.33
    c. 1.5
    d. 2.0

With no forwarding, we will have to stall 2 cycles for every instruction that needs data from the previous instruction. Since 50% need this, that means 50% of the instructions will take 2 cycles longer, or 3 cycles. So we had previously 1 instruction finishing per cycle and now 50% of the time we complete 1 every cycle and 30% of the time we get one every 3 cycles. So on average we get one every 1*0.5+3*0.5=2 cycles.

56. A processor is ideally pipelined to 20 stages, but due to hazards, 10% of the instructions are NOPs. What is the speedup over the original processor?
`2015-Aug-Hazards-6`
    a. 2x
    b. 10x
    c. 18x
    d. 20x

If the pipeline was full, an ideally pipelined processor with 20 stages would be 20x faster. But we waste 10% of our instructions, so we only realize 90% of the speedup, or 18x.

57. What would be a structural hazard?
`2014-Sep-Hazards-1`
    a. Needing a result that is in another part of the processor
    b. Needing a result that can't be computed because the hardware is busy
    c. Computing a result for an instruction that should not have been executed
    d. Any of the above

A result that is in another place is a data hazard. Computing a result that shouldn't be computed is a control hazard. Needing a result that can't be computed because the hardware is busy is a structural hazard. This can be solved by adding more hardware.

58. What would happen to performance if a pipeline could only read or write to the register file on each cycle?
`2014-Sep-Hazards-2`
    a. No change
    b. Better performance because the register file would be simpler and faster
    c. Worse performance because some instructions would have to stall
    d. Terrible performance because most instructions would have to stall

Most instructions read and write the register file, and these would have to stall for a cycle to get access to the register file. This would mean that the performance of the processor would be cut roughly in half as half of the pipeline would be filled with bubbles/stalls for the roughly half of instructions that need to read and write.

59. The bcopy routine does a bit copy from one memory location to another. The code consists of the following loop. What forwarding paths are needed in the standard 5-stage MIPS pipeline to avoid data hazards in this code?

```
loop:
addi R4, R4, 4          # increment source address
addi R5, R5, 4          # increment destination address
lw R6, 0(R4)            # load source data
sw R6, 0(R5)            # store data to destination
bne R4, R7, loop        # check if we are done with the copy
```

`2014-Sep-Hazards-3`
    a. WB-EX, MEM-EX
    b. WB-EX, WB-MEM
    c. MEM-EX, WB-MEM
    d. WB-EX, MEM-EX, WB-MEM

For both addi->lw we need to forward from the WB to the EX stage. For lw->sw we need to forward from the WB-MEM stage.

60. Which of the following did we not have to add/move to reduce the branch delay slots from 3 to 1?
`2014-Sep-Hazards-4`
    a. An extra adder for the PC
    b. An extra comparator for the branch true logic
    c. An extra register in the IF/ID pipeline register for the PC
    d. An extra MUX in the fetch stage to select the next PC

The IF/ID pipeline register already contained the PC in the original design since it had to be moved to the EX stage to compute the branch. We needed to add all the other hardware to these stages.

61. If a program has 20% branches with 1 branch delay slot per branch, and the compiler can find an instruction to put in the branch delay slot every other time, what is the performance slowdown due to the branch delay slot?
    2014-Sep-Hazards-5
    a. 5%
    b. 10%
    c. 20%
    d. 50%

    Half of the 20% of the time it takes an extra cycle: 10%*2 + 90%*1 = 1.1 cycles per instruction.

62. Are branch delay slots a good idea?
    2014-Sep-Hazards-6
    a. Yes, the compiler can find instructions to fill multiple slots
    b. Maybe, the tradeoff makes sense with simple processors and people can always recompile their code for new processors
    c. No, no one re-compiles software, so you have to emulate them in all future processors whether you need them or not
    d. It's not clear which one of these is most true

    Branch delay slots are clearly a bad idea. They buy a bit of performance in simple processors but become a headache later on when you have to emulate them in all future hardware and the compiler has to know about them.

63. What is needed to resolve all hazard in this example?
    ```
    Lw R1, 0(R2)
    Add R3, R1, R2
    Sw R3, 0(R2)
    ```
    2016-Oct-Hazards-1
    a. Forwarding from WB to MEM
    b. Forwarding from MEM to EX
    c. A and B
    d. Can't resolve; need to stall

    This cannot be completely solved by using forwarding. There are two hazards in this example, one between the first instruction and the second, and one between the second and third instruction. The second hazard can be resolved by forwarding the result of the addition from WB to MEM. However, the result of the first instruction is available only at the WB stage. Meanwhile, the next instruction, which needs the result of the load at the EX stage, is already in MEM stage. Therefore we need to stall.

64. What is needed to resolve all hazard in this example?
    ```
    add R4, R1, R2
    addi R5, R4, 42
    sw R5, 0(R6)
    ```
    2016-Oct-Hazards-2
    a. Forwarding from WB to MEM
    b. Forwarding from MEM to EX
    c. A and B
    d. Can't resolve; need to stall

    There are two hazards in this code. The first one is between the first and second instruction, because R4 is the source register for the second instruction and the destination register for the first instruction. This can be resolved by forwarding the result of the first instruction from the MEM stage to the EX stage.
    The second hazard is between the second and third instruction, due to the fact that the source register for the store instruction is the destination register of the second instruction (R5). This can be resolved by forwarding the result of the addi instruction from the WB stage to the MEM stage.

65. A processor is ideally pipelined to 10 stages, but due to hazards, 25% of the instructions are NOPs. What is the speedup over the original processor?
    2016-Oct-Hazards-3
    a. 10x
    b. 2.5x

c. 25x
d. 7.5

If the pipeline was full, an ideally pipelined processor with 10 stages would be 10x faster. But we waste 25% of our instructions, so we only realize 75% of the speedup, or 7.5x

66. How can forwarding help alleviate structural hazards?
`2016-Oct-Hazards-4`
    a. By forwarding a value to the previous stage
    b. By forwarding a value to the next stage
    c. A and B
    d. Forwarding cannot alleviate structural hazards

Structural hazards occur when the same hardware is needed by multiple instructions at the same time. Forwarding can only help with data hazards.

67. What logic does the forwarding unit need to forward from MEM to EX? (Rs, Rt are the sources and Rd is the destination)
`2016-Oct-Hazards-5`
    a. (MEM.Rt == EX.Rd) OR (MEM.Rt == EX.Rt)
    b. (MEM.Rd == EX.Rs) OR (MEM.Rd == EX.Rt)
    c. (MEM.Rs == EX.Rt) AND (MEM.Rs == EX.Rd)
    d. (MEM.Rd == EX.Rd) AND (MEM.Rd != EX.Rt)

Forwarding from MEM to EX occurs when the instruction in the MEM stage writes into a register that is one of the source registers of the instruction in the EX stage

68. When forwarding from MEM to EX, which signal from the Control unit do you need to inspect?
`2016-Oct-Hazards-6`
    a. EX.RegWrite
    b. EX.ALUop
    c. MEM.RegWrite
    d. MEM.MemtoReg

We need to check the MEM.RegWrite signal in order to find out whether the instruction in the MEM stage writes a register.

69. What kind of hazard problem does forwarding solve?
`2016-Jan-Hazards-1`
    a. Need data from an earlier instruction in the pipeline
    b. Need data from a later instruction in the pipeline
    c. Need logic that another instruction is using in the pipeline
    d. Need to execute the correct next instruction

Forwarding solves data hazards where the data is in the wrong place, however it must have already been computed.

70. Why can't we forward to avoid the branch delay slot?
`2016-Jan-Hazards-2`
    a. We can if we add a ID to IF forwarding path
    b. We can't forward the PC, only register values
    c. We can't detect the hazard in time
    d. We can't compute the correct next PC in time

The PC is computed at the end of the ID stage but we need it at the beginning of the IF stage. This means in the best case we get it one cycle to late.

71. What kind of hazard would an add instruction in a pipeline that executes two instructions at a time (dual-issue) have that it does not have in our standard (single-issue) pipeline? (Note that the dual-issue pipeline may have more of the same type of hazards, but we are looking for a type that the add instruciton would not have experienced before.)
`2016-Jan-Hazards-3`
    a. Control
    b. Data
    c. Structural

d. None

If two instructions can be executed at the same time you may need to adders to do them both at the same time. This is a structural hazard. The regular pipeline already has data hazards for the add instructions and adds do not have control hazards.

72. What forwarding is needed for this code?

```
add R3, R2, R0
lw R2, 0(R3)
sw R2, 0(R4)
```

2016-Jan-Hazards-4

a. MEM to EX
b. WB to EX
c. MEM to EX and WB to EX
d. MEM to EX and WB to MEM

The add to the lw needs forwarding from MEM to EX to calculate the address. The lw to sw needs forwarding from WB to MEM to store the data.

73. If a program has 10% branches with 2 branch delay slots per branch, and the compiler can fill 1 of the branch delay slots on average, what is the performance slowdown due to the branch delay slots?

2016-Jan-Hazards-5

a. 5%
b. 10%
c. 20%
d. 50%

10% of the time there is 1 extra NOP. So the cycles per instruction are 10%*2+90%*1=10%.

74. Are branch delay slots a good idea?

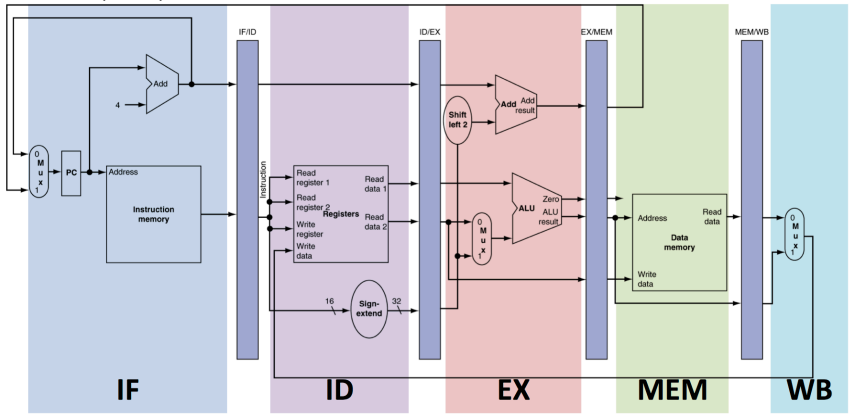
2016-Jan-Hazards-6

a. Yes: the compiler can put useful instructions in them almost all the time
b. Yes: people typically recompile their programs for new processors
c. A and B
d. No: all future processors have to emulate them to maintain compatibility

Compilers have a hard time filing too many branch delay slots and almost no one recompiles their software just because they get a new processor. The overhead of maintaining compatibility with branch delay slots for future processors makes them a bad decision..

75. What is needed to resolve the following hazard?

```
sub R4, R13, R14
add R14, R4, R13
```



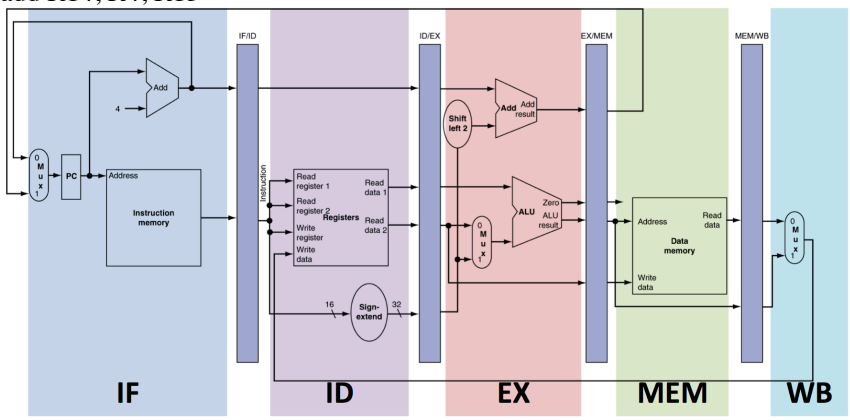
2013-Aug-Hazards-1

a. Forwarding from WB to EX
b. Forwarding from MEM to EX
c. Double-pumping the RF
d. Can't resolve: need to stall

The second instruction needs R4 in the EX stage, but the results of R4 from the previous instruction are only in the

MEM stage. So we need to forward from MEM to EX.

76. What is needed to resolve the following hazard?
lw R4, 12(R14)
add R14, R4, R13



2013-Aug-Hazards-2

a. Forwarding from WB to EX
b. Forwarding from MEM to EX
c. Double-pumping the RF
d. Can't resolve: need to stall

The second instruction needs R4 in the EX stage, but the results of R4 from the previous instruction are only available after the MEM stage. So we can't forward them (they don't exist yet) and have to stall.

77. Is stalling the pipeline a good way to solve hazards?

2013-Aug-Hazards-3

a. No, it hurts performance too much
b. No, it requires special compilers to insert stalls
c. A and B
d. None of the above

Stalls alone hurt performance too much. They don't require a compiler, however, as you can use the same logic we use to do forwarding to detect when to stall.

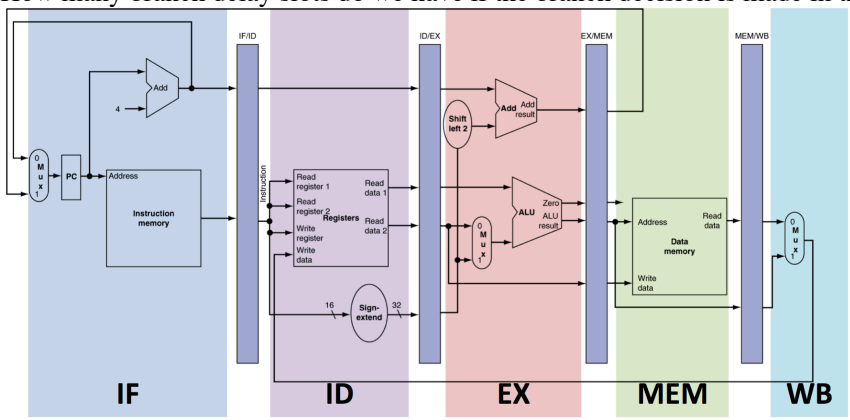
78. What does the data hazard detection logic do?

2013-Aug-Hazards-4

a. Look at the instructions in the different pipeline stages to see if they have conflicting register file accesses
b. Look at the instructions in the different pipeline stages to see if they write to the register file
c. A and B
d. None of the above

The data hazard detection logic looks at the instructions in the pipeline to see which registers they access and which ones are writing to the registers. From this it can determine what forwarding needs to happen.

79. How many branch delay slots do we have if the branch decision is made in the MEM stage?



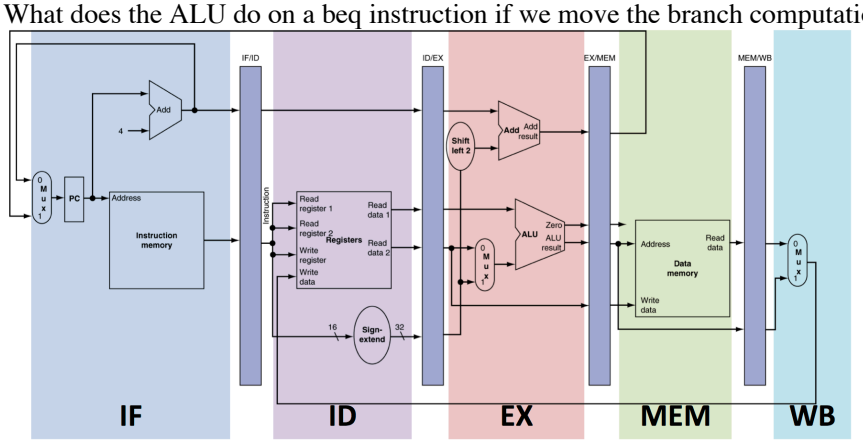
2013-Aug-Hazards-5

a. 0

b. 1
c. 2
d. 3

3: we need to wait for the IF, ID, and EX stages before we resolve a branch.

80. What does the ALU do on a beq instruction if we move the branch computation to the ID stage?



2013-Aug-Hazards-6

a. subtract to compute the jump condition
b. add to compute the jump condition
c. nothing
d. doesn't matter

When we move the branch computation earlier in the pipeline we add a separate comparator to determine if the registers are equal. This means we no longer need the ALU to do that comparison, so what it does is undefined and doesn't matter.
7. If 20% of the instructions are branches and we can only fill the branch delay slot 50% of the time, how much of a slowdown to we experience due to the branch delay slot?
a. 5%
b. 10%
c. 20%
d. 25%
20%*50% = 10% of the time we do an extra instruction, so we have a 10% slowdown.

81. What did double-pumping the register file fix?

2013-Apr-Hazards-1

a. The need to read the register file early for branches
b. The need to get data to the register file early for all dependent instructions
c. The need to read and write the register file at the same time
d. All of the above

Double pumping just allowed us to read and write in the same cycle from different pipeline stages. It did not fix all of the dependent instructions.

82. Why do we have problems with data hazards

2013-Apr-Hazards-2

a. Because writeback happens at the end of the pipeline
b. Because the ISA promises atomic execution
c. Because we only have one register file
d. Because the ISA promises sequential execution

The ISA promises that instructions will execute atomically (that is each one finishes before the next) but pipelining doesn't do that.

83. How many register values need to be forwarded in the following code on our standard 5-stage pipeline?

```
add R3, R4, R5
sub R2, R3, R4
beq R2, R3, done
```

2013-Apr-Hazards-3

a. 1

b. 2
c. 3
d. 4

R3 (add->sub), R2 (sub->beq), and R3 (add->beq)

84. Why do we prefer to forward data instead of stalling the processor?
`2013-Apr-Hazards-4`
    a. Forwarding is faster than stalling
    b. Forwarding logic is simpler than stall logic
    c. Stalling requires changing the compiler
    d. Stalling requires defining special bubble instructions

Forwarding makes the processor more complicated, but allows us to keep the pipeline more full by not having as many bubbles.

85. Why do we need a load delay slot here?
```
lw R7,12(R5)
sub R2, R7, R5
```
`2013-Apr-Hazards-5`
    a. The data from MEM is available at the end of the cycle but needed by EX at the start of the cycle
    b. The data from MEM may take much longer due to a cache miss
    c. The second instruction needs the data from first instruction in the cycle before it accesses the memory
    d. All of the above

The sub instruction needs the data in the EX stage at the beginning of the cycle, but the lw instruction only gets it back from the memory at end of the MEM stage.

86. Why can't we eliminate the branch delay slot by moving the branch computation further forward in the pipeline?
`2013-Apr-Hazards-6`
    a. We need to read the register file first, and that is in the ID stage
    b. We need to do a comparison, and that is in the EX stage
    c. We need to decode the instruction, and that is in the ID stage
    d. All of the above

We need to read the register file first to get the data to compare, and that doesn't happen until the ID stage.

8. Branch Prediction and Exceptions and Interrupts

1. Why do we predict branches in modern processors?
`2013-Dec-Branch Prediction and Exceptions and Interrupts-1`
    a. Can't afford the extra transistors to calculate the branch earlier
    b. The pipelines are long and we can't move the branch calculation earlier
    c. Branch predictors are faster than calculating the branch
    d. Predicting a branch is nearly perfectly accurate

We couldn't move the branch calculation early enough in the simple MIPS 5-stage pipeline because we needed to access the register file. Modern processors have even more complex and longer pipelines, which makes this harder. We could afford the transistors if we could actually build it.

2. What happens if two branch instructions share the same entry in the branch predictor?
`2013-Dec-Branch Prediction and Exceptions and Interrupts-2`
    a. The program crashes
    b. The branches will behave the same way
    c. The prediction accuracy is likely to be better
    d. The prediction accuracy is likely to be worse

If two branch instructions share the same branch predictor entry then that entry will try to learn what to do based on two branches. Unless they do exactly the same thing the prediction for one will be messed up by the data from the other. The predictions will not be the same since each branch will use the history of whatever came before it (including the other branch) to make a prediction.

3. For MIPS, making the branch delay slot part of the ISA was:
   2013-Dec-Branch Prediction and Exceptions and Interrupts-3
   a. Good: the compiler can fill them to do useful work
   b. Bad: newer processors have to pretend to have them even if they aren't 5-stages
   c. Unneeded: newer processors have branch predictors to keep going past a branch
   d. All of the above

   All of the above: they are good since the compiler can do something with that instruction slot, but it means the processor always has to do it correctly even if the pipeline changes. More fundamentally, branch predictors make this less of a problem because the processor just does the right thing. (most of the time.)

4. How many branches are correctly predicted with a 2-bit predictor initialized to strongly not taken?

```
for (i=0; i<100,000; i++) {
        if (i%2==1) { // branch 1: if i is odd
                // do something
        }
} // branch 2: loop
```

   2013-Dec-Branch Prediction and Exceptions and Interrupts-4
   a. branch 1: 50,000/100,000 branch 2: 99,997/100,000
   b. branch 1: 75,000/100,000 branch 2: 50,000/100,000
   c. branch 1: 50,000/100,000 branch 2: 99,998/100,000
   d. branch 1: 99,998/100,000 branch 2: 99,997/100,000

   Branch 1 will alternate TNTNTN because it happens whenever i is odd. The 2-bit predictor will therefore go between 00/01/00/01 but always predict not taken, which will mean it is 50% wrong. The outer loop will be wrong the first two times and the last time.

5. An application has 16% branches. On a particular processor, the branch predictor was correct 65% of the time. The processor has a 15 cycle penalty for each mispredicted branch and computes 1 instruction each cycle. What is the slowdown for this application due to mispredicted branches?
   2013-Dec-Branch Prediction and Exceptions and Interrupts-5
   a. 2.04x
   b. 1.94x
   c. 1.84x
   d. 0.84x

   (0.16 branches/instruction)*(0.35 misses/branch)*(15 cycles/miss)+(1 cycle/instruction) = 1.84 cycles/instruction = a slowdown of 1.84.

6. How is an exception or interrupt different from a procedure call?
   2013-Dec-Branch Prediction and Exceptions and Interrupts-6
   a. The program can't control where in the code it will happen
   b. The exception/interrupt determines what code is run
   c. Some exceptions or interrupts may switch to another program or the OS
   d. All of the above

   All of the above are true. Otherwise it is very similar: the exception or interrupt has to save any registers it might overwrite, but it can't count on the callee to have saved anything since the callee was interrupted!

7. What is the advantage of 2-bit branch predictors compared to 1-bit predictors?
   2017-Jan-Branch Prediction and Exceptions and Interrupts-1
   a. They change decision more slowly
   b. They learn faster
   c. They always make better predictions
   d. They use less logic

   When in the state "strongly taken" or "strongly not taken", it takes two consecutive mispredictions to change the next prediction. This is useful to reduce the number of mispredictions in the inner loop of two nested loops.

8. Branch prediction can solve control hazards by doing what?
   2017-Jan-Branch Prediction and Exceptions and Interrupts-2

a. Predicting the future
b. Forwarding the result
c. Replicating the branch calculation hardware
d. All of the above

The branch predictor predicts the branch before it happens.

9. What prediction strategy does our initial MIPS pipeline use?
2017-Jan-Branch Prediction and Exceptions and Interrupts-3
   a. No strategy
   b. Predict taken
   c. Predict not taken
   d. Randomly choose taken/not taken

The default MIPS pipeline always executes the instructions right after the branch. This is a predict not taken strategy.

10. Which is better: a processor with a 20-cycle branch penalty and a branch predictor that is 90% accurate or a processor with an 8-cycle branch penalty and a branch predictor that is 80% accurate?
2017-Jan-Branch Prediction and Exceptions and Interrupts-4
   a. 20-cycle/90%
   b. 8-cycle/80%
   c. Same
   d. Need more information

20*.1=2 and 8*.2=1.6. The first one has a 2 cycle overhead due to the branch misprediction while the second only has 1.6.

11. Consider the following code:

```
for (i = 0; i < 1000000; ++i)
{
    for (j = 0; j < 1000; ++j)
    {
        m[i][j] = 0
    } // branch j-loop
} // branch i-loop
```

How many mispredictions would we see if we ran this program on a CPU with a 1-bit branch predictor (initialized to "taken")?
2017-Jan-Branch Prediction and Exceptions and Interrupts-5
   a. 0
   b. 1 000 001
   c. 2 000 000
   d. 1 000 005

j-loop: wrong once in first j-loop, then twice per j-loop: 1 999 999 mispredictions
i-loop: wrong once at the very end: 1 misprediction.

12. A processor has a 2-bit branch predictor that can hold information on 128 different branches. A program has 129 branches. What will happen if we increase the size of the branch predictor table to hold information on 256 branches?
2017-Jan-Branch Prediction and Exceptions and Interrupts-6
   a. The program will stop crashing
   b. The program will stop producing the wrong results
   c. The program will run a lot faster
   d. The program will probably run a bit faster

Each branch will now be able to have its own predictor, so the program will probably be a bit faster. However, since we only had two branches that were conflicting before, it's unlikely to make a huge difference in performance.

13. Predictors help when?
2014-Apr-Branch Prediction and Exceptions and Interrupts-1
   a. If the answer is somewhere else in the processor

b. If the answer hasn't been computed yet
c. If multiple parts of the processor need the answer at the same time
d. All of the above

Predictors solve the problem that we don't know the answer yet.

14. What is the problem with predictors?
2014-Apr-Branch Prediction and Exceptions and Interrupts-2
    a. It's hard to predict the future with any accuracy
    b. We have to clean up if we make mistakes
    c. Predicting the future takes as long as computing it
    d. All of the above

It's actually really easy to predict the future most of the time. (Remember the branch predictor accuracy graphs we saw.) Predicting the future is also faster than waiting to see what happens. The problem is that if we are wrong we have to clean up the results of our misprediction.

15. Why do we need a branch target buffer (BTB)?
2014-Apr-Branch Prediction and Exceptions and Interrupts-3
    a. To predict whether the branch is taken or not
    b. To keep track of which branch the prediction is associated with
    c. To avoid having to wait for the branch address calculation
    d. To determine if the prediction is valid

The BTB keeps track of the target for a branch so that we don't have to wait for the branch address calculation to know where to go.

16. What percentage of the time will a 2-bit predictor that is initialized to strongly not-taken predict the if-branch correctly?

```
for (j=0; j<100,000,000; j++) {
        if ( j % 2 == 1) {
                // Do something
        }
}
```

2014-Apr-Branch Prediction and Exceptions and Interrupts-4
    a. 0%
    b. 1/100,000,000 (almost 0%)
    c. 50%
    d. 100%

This branch alternates between taken and not taken. (It does the "do something" every time j is odd.) Therefore a 2-bit branch predictor will move from strongly not taken to not taken but always predict not taken. As a result it will be wrong half the time.

17. Why is BTFN (backwards taken, forwards not-taken) a reasonable idea for a predictor?
2014-Apr-Branch Prediction and Exceptions and Interrupts-5
    a. It isn't
    b. Loops jump backwards and you mostly take those branches
    c. Loops jump forwards and you mostly don't take those branches
    d. Error checks are usually taken

BTFN relies on the idea that you usually execute loops many times and most loops contain branches that go backwards to go through the loop again.

18. If you have a branch predictor that keeps track of branch history based on the least significant bits of the PC and can store information for 16 separate branches, what happens when you have a program with the following behavior:
branch at 0010 1011 1010 1100 always not taken
branch at 0010 1011 1110 1100 always taken
2014-Apr-Branch Prediction and Exceptions and Interrupts-6
    a. The program crashes
    b. The branch predictor will only predict for one of the branches
    c. The branch predictor will confuse branches and predict incorrectly

d. All of the above

The predictor will use the same bits 1011 from both of the branches to keep track of branch behavior and as a result it won't be able to tell them apart. This will lead to a bad prediction, but it won't cause the computer to crash.

19. Why are branch predictors useful?
`2015-Oct-Branch Prediction and Exceptions and Interrupts-1`
    a. They allow us to only execute the needed instructions when they are correct
    b. They allow us to squash instructions that should never have entered the pipeline
    c. They enable forwarding between pipeline stages to avoid branch dependencies
    d. All of the above

Branch predictors try to predict the outcome of branches. Without a branch predictor, the CPU would either have to stall until the outcome of the branch is known, or start executing instructions and "squash" them if they should not have been executed.

20. What is the advantage of 2-bit branch predictors compared to 1-bit predictors?
`2015-Oct-Branch Prediction and Exceptions and Interrupts-2`
    a. They change decision more slowly
    b. They learn faster
    c. They always make better predictions
    d. They use less logic

When in the state "strongly taken" or "strongly not taken", it takes two consecutive mispredictions to change the next prediction. This is useful to reduce the number of mispredictions in the inner loop of two nested loops.

21. A program contains 1 million instructions, of which 15% are branches. If we run it on a CPU with a 20 cycles branch penalty and a branch predictor that is 85% accurate, how many cycles will be wasted due to mispredictions?
`2015-Oct-Branch Prediction and Exceptions and Interrupts-3`
    a. 100 000 cycles
    b. 200 000 cycles
    c. 450 000 cycles
    d. 1 000 000 cycles

This program contains 150 000 branches. 15% of them will be mispredicted, and each misprediction costs 20 cycles. 150 000 * 0.15 * 20 = 450 000 cycles wasted.

22. How would the performance change if we ran the program described in the previous question on a CPU with a 30 cycles branch penalty but with a branch predictor that is 90% accurate?
`2015-Oct-Branch Prediction and Exceptions and Interrupts-4`
    a. Need more information
    b. The performance would improve
    c. The performance would decrease
    d. The performance would remain the same

The program still contains 150 000 branches. 10% of them will be mispredicted, and each misprediction costs 30 cycles. 150 000 * 0.10 * 30 = 450 000 cycles wasted, so the performance remains the same compared to the previous predictor.

23. Consider the following code:

```
for (i = 0; i < 1000000; ++i)
{
   for (j = 0; j < 1000; ++j)
   {
      m[i][j] = 0
    } // branch j-loop
} // branch i-loop
```

How many mispredictions would we see if we ran this program on a CPU with a 1-bit branch predictor (initialized to "taken")?
`2015-Oct-Branch Prediction and Exceptions and Interrupts-5`
    a. 0
    b. 1 000 001

c. 2 000 000
d. 1 000 005

j-loop: wrong once in first j-loop, then twice per j-loop: 1 999 999 mispredictions
i-loop: wrong once at the very end: 1 misprediction.

24. What if we ran the following code:

```
for (i = 0; i < 1000000; ++i)
{
   for (j = 0; j < 1000; ++j)
   {
      m[i][j] = 0
    } // branch j-loop
} // branch i-loop
```

on a CPU with a 2-bit branch predictor (initialized to "strongly taken")?
`2015-Oct-Branch Prediction and Exceptions and Interrupts-6`

a. 0
b. 1 000 001
c. 2 000 000
d. 1 000 005

j-loop: wrong once per j-loop: 1 000 000 mispredictions
i-loop: wrong once at the very end: 1 misprediction.

25. How accurate is the branch predictor in your computer's processor?
`2018-Oct-Branch Prediction and Exceptions and Interrupts-1`

a. Less than 20% correct predictions
b. Around 50% correct predictions
c. More than 90% correct predictions, but not perfect
d. 100% perfect

Branch prediction is essential for getting good performance from long pipelines. Today's processors have well over 90% accuracy (typically 98%+) but are not perfect.

26. What does the Branch Target Buffer (BTB) do?
`2018-Oct-Branch Prediction and Exceptions and Interrupts-2`

a. The BTB predicts whether the branch is taken or not
b. The BTB prevents us from having to wait for the branch address calculation
c. The BTB keeps track of the history of branches
d. The BTB determins if the branch prediction is a good prediction

The Branch Target Buffer keeps track of the address the branch went to last time to enable the processor to jump there on a prediction without having to wait for the address calculation.

27. When do predictors help?
`2016-Aug-Branch Prediction and Exceptions and Interrupts-1`

a. If the answer is somewhere else in the processor
b. If multiple parts of the processor need to do the same type of operation at the same time
c. If the answer can't be computed yet
d. All of the above

Predictors solve the problem where we can't compute the answer early enough.

28. Why are incorrect predictions a problem?
`2016-Aug-Branch Prediction and Exceptions and Interrupts-2`

a. We execute the wrong instructions and have to clean up the register file
b. We execute the wrong instructions and have to squash them
c. We update the branch predictor with the wrong result
d. All of the above

The instructions in the pipeline do not get to write back to the register file before we know the branch outcome and can squash them. We only update the branch predictor when we know the true outcome of a branch.

29. A processor takes 5 cycles to compute the outcome of a branch. What happens to those 5 pipeline slots on a processor with a branch predictor?

2016-Aug-Branch Prediction and Exceptions and Interrupts-3

a. The compiler tries to fill them with useful instructions
b. The processor fills them with the next 5 PCs after the branch
c. The processor fills them with the next 5 PCs based on the branch predictor
d. The processor fills them with NOPs if it can't predict, and instructions otherwise

The branch predictor predicts the next PC from the branch and then the processor fills those slots with the next 5 instructions on that path.

30. Which processor would you rather have assuming the pipelining speedup is only limited by branch mispredictions? Processor A with a 10 cycle pipeline, a 5 cycle branch misprediction penalty, and a 95% accurate branch predictor, or Processor B with a 50 cycle pipeline, a 25 cycle branch misprediction penalty, and a 99% accurate predictor?

2016-Aug-Branch Prediction and Exceptions and Interrupts-4

a. A
b. B
c. They're both just as good
d. Need more information

For each branch, B will give you a penalty of 25*1%=0.25 and A will give you a penalty of 5*5%=0.25. However, B will have a 50x pipeline speedup while A will only have a 10x pipeline speedup. Both will be limited by mispredictions, but since they have the same penalty this will affect them in the same way, and B's much higher performance will win.

31. Which predictor will not have 50% accuracy for the if statement in this loop?

```
loop {
  i++
  if (is_odd(i))
   b=b*2;
}
```

2016-Aug-Branch Prediction and Exceptions and Interrupts-5

a. 1-bit predictor
b. 2-bit predictor
c. BTFNT
d. Static

The one-bit predictor will have 0% accuracy as it will always predict the last value which is wrong. The others will always predict one value which will be right half the time.

32. The following program has an interesting characteristic: branch A can be predicted nearly perfectly, and 90% of the time A is taken B is also taken. By itself B appears to be random. How could you build a branch predictor that would do a better job of predicting the B branch?

```
if (A)
   do_something
if (B)
   do_something
```

2016-Aug-Branch Prediction and Exceptions and Interrupts-6

a. Use more bits in the predictor
b. Use the PCs of the past two branches to look up a prediction
c. Use the outcomes of the past two branches to look up a prediction
d. Use the PCs and outcomes of the past two branches to look up a prediction

If you use the PCs and outcomes of the past two branches then you will be able to find the case where A was taken and accurately predict B will be taken.

33. Which of the following statements is true about branch prediction?

2014-Dec-Branch Prediction and Exceptions and Interrupts-1

a. Modern processors run too fast to stall the pipeline or kill instructions, so we have no choice but to predict branches.

b. The performance penalty for killing instructions on the wrong branch or stalling to wait for a branch to be resolved are so high that branch prediction is essential.
c. Compilers today are good enough to fill many branch delay slots, but it takes them too long to compile with this option enabled, so we need branch prediction
d. All of the above

Processor pipelines are 15+ instructions deep and compilers are not good enough to find that many independent instructions for each branch. We do still need to be able to stall the processor pipeline or kill instructions if we misspredict.

34. What is the slowdown for processors that does one instruction per cycle and has a branch predictor that is 85% accurate with a 75-cycle mis-prediction cost? (Assume 35% of instructions are branches).

2014-Dec-Branch Prediction and Exceptions and Interrupts-2

a. ~4x faster
b. ~2x slower
c. ~5x slower
d. ~7.5x slower

35% of the time we will have a 1-85%= 15% chance of paying a 65 cycle penalty in addition to the 1 cycle it normally takes.
(0.35 branches/instruction)*(0.15 mispredictions/branch)*(75 cycles/misprediction) + 1 cycle/instruction = 4.93 cycles per instruction, or 5x slower

35. A program with 17 branches is executed on a processor using a 16-entry branch predictor. What would happen?

2014-Dec-Branch Prediction and Exceptions and Interrupts-3

a. The program will crash
b. The predictor will use the stack to save the 17th branch
c. The predictor will confuse two of their predictions
d. The predictor will miss miss-predict one of them

The predictor will put the results of two of the branches into the same place, which will cause it to confuse the two of them and probably result in a far worse prediction.

36. A 1-bit branch predictor has 4 entries and is direct-mapped based on the PC of the branch. What will happen with second branch in this loop?

```
start:
  beq R1, R2, loop
    jal update
    j start
loop:
  add R3, R1, R2
  beq R1, R2, start
```

2014-Dec-Branch Prediction and Exceptions and Interrupts-4

a. Predict fine
b. Predict badly
c. Depends on the initial values of R1 and R2
d. Depends on how update changes R1 and R2

The prediction will be fine here because both branches do the same thing, despite mapping to the same entry. The two branches are exactly 4 instructions apart so they will map to the same entry in the branch predictor. However, if R1 equals R2 ten we jump to loop and we check if they are equal again. Since the branch predictor will store the prediction for both in the same place, we will perfectly predict what to do with the second branch because we learned it from the first branch.

37. A processor has two branch predictors (1-bit and BTFNT). It uses a third predictor to learn which one to use by first trying 1-bit and then switching to BTFNT if it would work better. (This is called a tournament predictor.) The 1-bit predictor starts in the not-taken state. What will the third predictor choose for the k- and j-branches below to get the fewest mispredictions?

```
For (k=0; k<100,000; k++) {
        For (j=0; j<3; j++) {
        // Do something
        } // j branch
```

```
} // k branch
```

2014-Dec-Branch Prediction and Exceptions and Interrupts-5

a. j: 1 bit, k: 1 bit
b. j: 1 bit, k: BTFNT
c. j: BTFNT, k: 1 bit
d. j: BTFNT, k: BTFNT

The j-loop with the 1-bit predictor will be NT(wrong), T(right), T(wrong), NT(wrong), T(right), T(wrong), so wrong 2/3 times. The j-loop with BTFNT will be T(right), T(right), T(wrong) right 2/3 times. So the j-loop will use BTFNT. The k-loop is the same for both, but the predictor starts with 1-bit and only switches if BTFNT is better, so it will stay with 1-bit.

38. A pattern-based branch predictor stores two values for each branch: the pattern and the current location in the pattern. How many bits of storage would this predictor need to store to correctly predict this pattern of branches: NTTNTTNTTNTT?

2014-Dec-Branch Prediction and Exceptions and Interrupts-6

a. 2
b. 3
c. 4
d. More

We need 3 bits of pattern since the pattern repeats every three branches plus 2 bits of storage to tell where we are in the pattern.

39. What happens if you have fewer entries in your branch predictor than you have branches in your program?

2015-Apr-Branch Prediction and Exceptions and Interrupts-1

a. The branch predictor can't make a prediction so it will turn off
b. The branch predictor will always predict incorrectly
c. The branch predictor will be very likely to predict less accurately
d. You won't be able to run your program

Multiple branch addresses will point to the same entry in the branch predictor (just like multiple addresses point to the same place in a cache) and they will share predictors, which is very likely to result in worse predictions.

40. Which processor would you rather have, assuming the other characteristics of the processor are the same? Processor A with a 100 cycle branch misprediction penalty and a 99.9% accurate branch predictor or Processor B with a 10 cycle branch misprediction penalty and a 99% accurate branch predictor?

2015-Apr-Branch Prediction and Exceptions and Interrupts-2

a. A
b. B
c. They're both just as good
d. Need more information

For each branch, A will give you a penalty of 100*.001=0.1 and B will give you 10*.01=0.1. You don't need to know the branch frequency since it would have the same impact on both.

41. Which processor would you rather have if the pipelining speedup is only limited by branch mispredictions? Processor A with a 100 cycle pipeline, a 100 cycle branch misprediction penalty and a 99.9% accurate branch predictor or Processor B with a 10 cycle pipeline, a 10 cycle branch misprediction penalty and a 99% accurate branch predictor?

2015-Apr-Branch Prediction and Exceptions and Interrupts-3

a. A
b. B
c. They're both just as good
d. Need more information

For each branch, A will give you a penalty of 100*.001=0.1 and B will give you 10*.01=0.1. However, A will have a 100x pipelining speedup and B will only have a 10x speedup. Both will be limited because of branch mispredictions, but since they have the same penalty, this will affect them in the same way.

42. For which kind of application would a BTFN branch predictor work best?

2015-Apr-Branch Prediction and Exceptions and Interrupts-4

a. Applications with plenty of code that jumps forward over error handling code

b. Applications with plenty of loops that are usually taken backwards
c. Applications with loops that start out being taken all the time and then switch to not being taken later in the program
d. All of the above

Loops are the most common code and they typically go backwards.

43. Why is a 2-bit predictor often better than a 1-bit predictor?
`2015-Apr-Branch Prediction and Exceptions and Interrupts-5`
a. It learns faster
b. It changes more slowly
c. It uses less logic
d. It predicts more accurately

The 2-bit predictor can be better because it changes its prediction more slowly, so it is less likely to be confused by an occasional misprediction.

44. If an interrupt occurs which causes the operating system to switch to another program. Whose responsibility is it to save the current program's registers to the stack?
`2015-Apr-Branch Prediction and Exceptions and Interrupts-6`
a. The current program
b. The caller
c. Don't need to
d. The operating system

The current program doesn't know it is being interrupted (since it can't control interrupts) so it wouldn't know what registers to save. The caller is part of the current program. The new program doesn't know what the other program is doing either. Only the operating system, which handles the program switch, knows to save and restore the program's registers.

45. What can we do to avoid the penalty of the branch delay slot?
`2013-Jan-Branch Prediction and Exceptions and Interrupts-1`
a. Double-pump the register file
b. Re-order instructions
c. Put in a bubble
d. Move the branch logic forward

46. What is the slowdown due to a branch predictor that has 90% accuracy on the Intel Nehalem processor with a 17 cycle branch misprediction cost and a normal CPI of 1.0 for a program with 25% branches?
`2013-Jan-Branch Prediction and Exceptions and Interrupts-2`
a. 0.4
b. 1.4
c. 3.8
d. 4.8

CPI = 1.0+17*.1*.25=1.425.

47. How many mispredictions does the BTFNT predictor make for this loop?

```
k=0
loop:
  k++
  branch (k>=10) to skip
    j++
  skip:
branch if (k<=1000) to loop
```

`2013-Jan-Branch Prediction and Exceptions and Interrupts-3`
a. 1
b. 11
c. 991
d. 1010

48. Why do we need a valid bit but no dirty bit in the branch target buffer?
`2013-Jan-Branch Prediction and Exceptions and Interrupts-4`
    a. Branch targets can be invalid, but we never write to the BTB
    b. Branch targets can be invalid, but we never write the BTB back to DRAM
    c. Branch targets can not be invalid, so we always write them back
    d. We don't: we need both a dirty bit and a valid bit in the BTB

49. Was the decision to have a branch delay slot in MIPS a good one?
`2013-Jan-Branch Prediction and Exceptions and Interrupts-5`
    a. Yes, because it enabled early MIPS processors to be easier to build
    b. Yes, because compiler technology can easily fill it
    c. No, because newer MIPS processors have to pretend to have just one
    d. No, because compilers can't reliably fill it

50. How many exceptions could you have at the same time in a 5-stage MIPS pipeline?
`2013-Jan-Branch Prediction and Exceptions and Interrupts-6`
    a. 2
    b. 3
    c. 4
    d. 5

51. Why do we need branch predictors on modern processors?
`2015-Aug-Branch Prediction and Exceptions and Interrupts-1`
    a. The pipelines are so long that we could never fill all the branch delay slots
    b. The pipeline is too long to kill instructions from a wrong branch
    c. The processor runs too fast to stall the pipeline
    d. All of the above

    Processor pipelines are 15+ instructions deep and we can never find that many instructions for each branch. We still have to be able to kill instructions from the wrong branch and stall the processor.

52. When do predictors help avoid hazards?
`2015-Aug-Branch Prediction and Exceptions and Interrupts-2`
    a. When the answer hasn't been computed yet
    b. When multiple instructions need the same hardware at the same time
    c. When the data is in another stage
    d. All of the above

    Predictors solve the problem when we need data that hasn't been computed yet.

53. Which is better: a processor with a 20-cycle branch penalty and a branch predictor that is 90% accurate or a processor with an 8-cycle branch penalty and a branch predictor that is 80% accurate?
`2015-Aug-Branch Prediction and Exceptions and Interrupts-3`
    a. 20-cycle/90%
    b. 8-cycle/80%
    c. Same
    d. Need more information

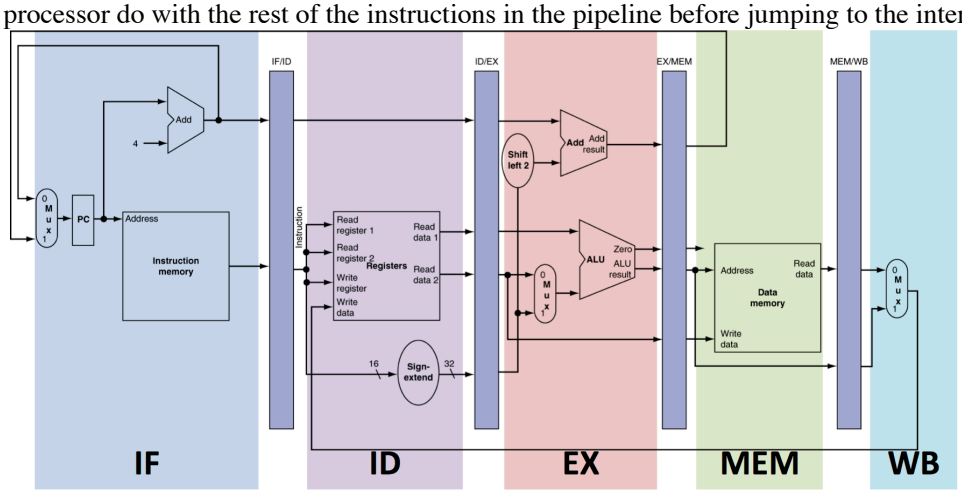
    20*.1=2 and 8*.2=1.6. The first one has a 2 cycle overhead due to the branch misprediction while the second only has 1.6.

54. A processor has a 2-bit branch predictor that can hold information on 128 different branches. A program has 129 branches. What will happen if we increase the size of the branch predictor table to hold information on 256 branches?
`2015-Aug-Branch Prediction and Exceptions and Interrupts-4`
    a. The program will stop crashing
    b. The program will stop producing the wrong results
    c. The program will run a lot faster
    d. The program will probably run a bit faster

    Each branch will now be able to have its own predictor, so the program will probably be a bit faster. However, since we only had two branches that were conflicting before, it's unlikely to make a huge difference in performance.

55. A divide-by-zero exception in the EX stage causes an interrupt that jumps to code to clean up the math. What should the

processor do with the rest of the instructions in the pipeline before jumping to the interrupt?



2015-Aug-Branch Prediction and Exceptions and Interrupts-5

a. IF-finish, ID-finish, EX-finish, MEM-finish, WB-finish
b. IF-kill, ID-kill, EX-finish, MEM-finish, WB-finish
c. IF-kill, ID-kill, EX-kill, MEM-finish, WB-finish
d. IF-kill, ID-kill, EX-kill, MEM-kill, WB-kill

The instruction that caused the exception needs to be killed (so we can fix the exception) and all the instructions that came after it. This means killing EX/ID/IF. The instructions that were before it should finish so we can restart at the instruction with the exception after cleaning up..

56. Which of the following is true for both a procedure call and an interrupt?

2015-Aug-Branch Prediction and Exceptions and Interrupts-6

a. Can control where in the code it will happen
b. Have to save and restore registers that are overwritten
c. Can determine what code is executed
d. Know which program will execute the code

An interrupt can happen at any place in the code and can jump to code outside of the program, and, indeed, switch to the OS or another program. However, both interrupts and procedure calls need to make sure they don't corrupt the rest of the running code's registers.

57. Branch prediction can solve control hazards by doing what?

2014-Sep-Branch Prediction and Exceptions and Interrupts-1

a. Predicting the future
b. Forwarding the result
c. Replicating the branch calculation hardware
d. All of the above

The branch predictor predicts the branch before it happens.

58. What prediction strategy does our initial MIPS pipeline use?

2014-Sep-Branch Prediction and Exceptions and Interrupts-2

a. No strategy
b. Predict taken
c. Predict not taken
d. Randomly choose taken/not taken

The default MIPS pipeline always executes the instructions right after the branch. This is a predict not taken strategy.

59. What do you have to do to instructions in the pipeline that are executed due to a misspredicition?

2014-Sep-Branch Prediction and Exceptions and Interrupts-3

a. Nothing if they are memory instructions
b. Prevent them from changing anything
c. Insert instructions to reverse their effects
d. All of the above

They need to be prevented from changing anything. That is, they must not write to the register file, memory, or PC. If their writes are disabled then they effectively don't do anything.

60. Which statement is true?
`2014-Sep-Branch Prediction and Exceptions and Interrupts-4`
    a. Dynamic predictors can adapt to program behavior
    b. Static predictors rely on the compiler to know what branches will do
    c. Dynamic predictors can behave worse than static predictors for some code
    d. All of the above

Dynamic predictors learn program behavior and adapt. Static predictors require that the compiler know what is likely to happen with a branch when the program is compiled A 1-bit dynamic predictor can always predict the wrong branch on a loop that changes on every iteration, while a static predictor would have been right every other time.

61. A processor has a 1000-entry dynamic branch predictor table. What happens for a program that has 1500 branches?
`2014-Sep-Branch Prediction and Exceptions and Interrupts-5`
    a. Roughly 500 of the branches will not get branch predictions
    b. Roughly 1000 of the branches will mess up each other's predictions
    c. The program will crash
    d. The operating system will have to clean up the misspredicted branches

1000 of the branches (roughly) will share predictors with another branch. This means that the single shared predictor will be trying to learn the behavior of two different branches, which will mess up the predictions.

62. If we add a branch target buffer (BTB) to the branch predictor, do we still need an adder to calculate the branch address?
`2014-Sep-Branch Prediction and Exceptions and Interrupts-6`
    a. Yes, but only if we predict the branch incorrectly
    b. Yes, to verify that the target was correct even if we predict correctly
    c. Only the first time we see the branch so we can store the address in the BTB
    d. No, the instruction provides the address

We still need to calculate the address to know if the address in the BTB was correct. This is particularly true for instructions where the address is not a constant, such as JR.

63. Which of the following is a reasonable way to handle an instruction we execute for a mispredicted branch that we should not have executed?
`2016-Oct-Branch Prediction and Exceptions and Interrupts-1`
    a. Replace it with a NOP instruction
    b. Replace it with an ADD R0, R1, R2 instruction
    c. Change its pipeline registers to MemWrite=0, RegWrite=0, and PCSrc=PC+4
    d. All of the above

All of these will prevent the instruction from changing the processor state: they will no longer change the register file, the memory, or the PC.

64. What was the MIPS branch prediction strategy before we added a branch predictor?
`2016-Oct-Branch Prediction and Exceptions and Interrupts-2`
    a. Predict taken
    b. Predict not taken
    c. No strategy
    d. Random

The default MIPS pipeline we had always executed the instructions after the branch before it was resolved. This is a predict not-taken strategy.

65. Which statement is false?
`2016-Oct-Branch Prediction and Exceptions and Interrupts-3`
    a. Static predictors can adapt to program behavior
    b. Dynamic predictors rely on the compiler to know what branches will do
    c. Dynamic predictors are always better than static predictors
    d. All of the above

Static predictors do not adapt: they are set by the compiler. Dynamic predictors can adapt, but they are not necessarily better than static ones.

66. A processor's branch predictor has 100 entries. What happens for a long-running program that uniformly uses 200 branches?
`2016-Oct-Branch Prediction and Exceptions and Interrupts-4`
    a. Half of the branches will not get predictions
    b. The processor will crash
    c. The branches will mess up each other's predictions
    d. The predictor will work well for all branches

The branch predictor tracks only 100 branches, so when it gets to a branch that it is not tracking it will reuse one of the slots to track it, which will mess up the prediction for the other branch that was there previously.

67. A processor's branch predictor has 100 entries. What happens for a long-running program that has 200 branches, but only uses 100 for the first half of the program and the other 100 in the second half?
`2016-Oct-Branch Prediction and Exceptions and Interrupts-5`
    a. Half of the branches will not get predictions
    b. The processor will crash
    c. The branches will mess up each other's predictions
    d. The predictor will work well for all branches

During the first half of the program the first 100 branches will fit into the predictor and get good predictions. During the second half the second 100 will. In-between there will be a short time when the new branches are learned, but overall it will work well for all branches.

68. What is the slowdown for an otherwise perfect 10-stage pipelined processor that has 2 branch delay slots, a branch predictor that is 90% accurate, and a program that has 10% branches?
`2016-Oct-Branch Prediction and Exceptions and Interrupts-6`
    a. 1%
    b. 2%
    c. 10%
    d. 20%

10% of the time we have branches, and 10% of that they cost 3 cycles. So we have 10%*10%*3+99%*1 = CPI of 1.02, or 2% slowdown.

69. How do pipelines complicate exceptions?
`2016-Jan-Branch Prediction and Exceptions and Interrupts-1`
    a. They don't: exceptions and interrupts just cause the program to jump to another piece of code
    b. We can't squash instructions that are partly completed depending on where they are in the pipeline
    c. Exceptions from later instructions may come before exceptions from earlier instructions depending on where they are in the pipeline
    d. B and C

Squashing instructions is certainly possible as we saw with hazards. The problem with exceptions in a pipeline is that there may be instructions that come later and generate an exception in an earlier pipeline stage before earlier instructions generate their exception in a later stage. If the processor gets the exceptions out of order it may cause incorrect execution.

70. What does the processor do if it detects that a branch was predicted incorrectly?
`2016-Jan-Branch Prediction and Exceptions and Interrupts-2`
    a. Squash the instructions that followed the incorrect prediction
    b. Set the next PC to the correct branch result
    c. Update the predictor
    d. All of the above

If a branch is predicted incorrectly the processor has to clean up (squash the instructions that should not have been executed), restart at the right point (next PC), and update the predictor to do a better job next time.

71. A branch predictor is like a cache in that it stores predictions for many different branches. How does the branch predictor look up the correct prediction for a given branch?
`2016-Jan-Branch Prediction and Exceptions and Interrupts-3`

a. It uses the result of the last branch to look up the prediction
b. It uses the result of the branch to look up the prediction
c. It uses the PC of the branch to look up the prediction
d. It uses the type of instruction to look up the prediction

The branch predictor uses the PC of the branch to look up the prediction data for a branch, and then both updates that based on the results of the branch and uses it to predict whether or not the branch will be taken.

72. A tournament branch predictor uses a predictor to predict which of two other predictions to use. (E.g., predictor C chooses between the predictions of predictors A and B.) When is this a better design than a single predictor?
`2016-Jan-Branch Prediction and Exceptions and Interrupts-4`
    a. When branches are largely random
    b. When some branches are predicted best by different predictors
    c. When static predictors aren't good enough
    d. When branches switch between taken and not taken

The tournament predictor allows the computer to learn which kind of predictor works best for a given branch. This is helpful if some branches are best predicted by different types of predictors. Random branches can't be predicted and most dynamic predictors are better than static ones. If a branch switches between taken and not taken the predictor has to re-learn regardless.

73. Why do we have a BTB (branch target buffer) in addition to a branch predictor?
`2016-Jan-Branch Prediction and Exceptions and Interrupts-5`
    a. To avoid having to wait for the branch address calculation
    b. To predict the next PC for non-branch instructions
    c. To eliminate the branch delay slot
    d. To predict whether JR instructions are taken

The BTB stores the instruction to branch to so that we can branch without having to wait for the branch target calculation.

74. Which is better for a 1 IPC processor: a 5-stage pipeline (2 cycle misprediction penalty) with a 98% accurate branch predictor or a 7-stage pipeline (4 cycle misprediction penalty) with a 99% accurate branch predictor?
`2016-Jan-Branch Prediction and Exceptions and Interrupts-6`
    a. 5-stage
    b. 7-stage
    c. Same
    d. Need more information

5-stage: 2%*2+98%*1=1.02 CPI. 7-stage: 1%*4+99%*1=1.03 CPI.

75. Why do we need branch prediction on modern processors?
`2013-Aug-Branch Prediction and Exceptions and Interrupts-1`
    a. The processor runs too fast to stall the pipeline
    b. The pipelines are so long that we could never fill all the branch delay slots
    c. The pipeline is too long to kill instructions from a wrong branch
    d. All of the above

Processor pipelines are 15+ instructions deep and we can never find that may instructions for each branch. Either way we have to be able to kill instructions from the wrong branch and stall the pipeline.

76. What is the slowdown for processor that does one instruction per cycle and has a branch predictor that is 95% accurate with a 50 cycle misprediction cost? (Assume 20% of instructions are branches.)
`2013-Aug-Branch Prediction and Exceptions and Interrupts-2`
    a. 2x faster
    b. 1.5x slower
    c. 2.5x slower
    d. 3.5x slower

20% of the time we will have a 1-95%=5% chance of paying a 50 cycle penalty in addition to the 1 cycle it normally takes. (0.2 branches/instruction)*(0.05 mispreditions/branch)*(50 cycles/misprediction)+(1 cycle/instruction) = 1.5 cycles per instruction, or 1.5x slower.

77. What is the reasoning behind the BTFN branch predictor?
2013-Aug-Branch Prediction and Exceptions and Interrupts-3
    a. Loops are the most common taken branches and they are usually backwards
    b. Error code usually has forward jumps and are not frequently taken
    c. We guess 50% each way
    d. All of the above

Loops are the most common code and they typically go backwards.

78. What happens if you have a branch predictor with 8 entries, but your program has 9 branches?
2013-Aug-Branch Prediction and Exceptions and Interrupts-4
    a. The predictor miss-predicts one of them
    b. The predictor confuses two of them
    c. The predictor puts one on the stack to hold the 9th branch
    d. The program crashes

The predictor will put the results of two of the branches into the same place, which will cause it to confuse the two of them and probably result in a far worse prediction.

79. Why n-bit branch predictors better than 1-bit branch predicors?
2013-Aug-Branch Prediction and Exceptions and Interrupts-5
    a. You can make more miss-predictions before forgetting your previous prediction
    b. You can predict for more branches
    c. You can predict more complicated patterns
    d. All of the above

The larger number of bits allows you to keep your prediction through more miss-predictions.

80. When an exception occurs, what happens to the other instructions in the pipeline?
2013-Aug-Branch Prediction and Exceptions and Interrupts-6
    a. They are all killed
    b. They are killed if they come after the instruction with the exception
    c. They are killed if they have a data dependency on the instruction with the exception
    d. The operating system decides which instructions to kill

The instructions after the instruction with the exception have to be killed because we need to handle the exception before continuing.

81. What is the effective CPI (what you actually get) of a processor where each instruction takes 1 cycle and it has 2 branch delay slots, 20% of the instructions are branches, and the branch delay slots can be filled only 50% of the time?
2013-Apr-Branch Prediction and Exceptions and Interrupts-1
    a. 1.1
    b. 1.2
    c. 1.4
    d. 1.5

20% of the time we will have to have 2 branch delay slots, but 50% of that time we can use them. That means we'll see a CPI increase of 20%*(50%*2) = 0.2, for a total CPI of 1.2.

82. What happens if you have more branches in your program than you have spaces in your branch predictor?
2013-Apr-Branch Prediction and Exceptions and Interrupts-2
    a. Can't make a prediction
    b. Can't run your program
    c. May predict incorrectly
    d. Always predict incorrectly

Multiple branch addresses will point to the same entry in the branch predictor (just like multiple addresses point to the same place in a cache) and they will share predictors.

83. What is the benefit of a 2-bit predictor over a 1-bit predictor?
2013-Apr-Branch Prediction and Exceptions and Interrupts-3
    a. Faster

b. Predicts backwards jumps better
c. Doesn't get fooled by a single wrong prediction
d. Always works better

2-bits is only better because it allows you to have "weak" states, which mean you can have a single wrong prediction before changing your prediction.

84. Which is better: a processor with a 16-cycle branch penalty and a branch predictor that is 90% accurate or a processor with an 8-cycle branch penalty and a branch predictor that is 80% accurate? (assume you can otherwise keep the pipeline full.)
`2013-Apr-Branch Prediction and Exceptions and Interrupts-4`
    a. 16-cycle/90% accurate
    b. 8-cycel/80% accurate
    c. Same
    d. Can't tell without knowing the percentage of branches

    16*.1=1.6, 8*0.2=1.6 - both have a 1.6 cycle expected overhead for a branch, so they will both be the same.

85. What decides to jump to the interrupt handler code on an interrupt?
`2013-Apr-Branch Prediction and Exceptions and Interrupts-5`
    a. The operating system
    b. The user code
    c. The processor
    d. The I/O device

    The processor receives the signal that the I/O device is ready and then causes the jump to the right location to handle it.

86. What happens to later instructions in the pipeline when an exception occurs?
`2013-Apr-Branch Prediction and Exceptions and Interrupts-6`
    a. They are paused until after the exception is handled
    b. They are killed
    c. They continue
    d. They continue unless they also have an exception

    The instructions are killed because the first one needs to be handled before the processing can resume.

9. Input/Output

1. What are the tradeoffs between polling and interrupts?
`2013-Dec-Input/Output-1`
    a. Polling uses up little processor time but has a high latency
    b. Interrupts use up a lot of processor time but have a low latency
    c. Interrupts are good for low-latency I/O and polling for high-throughput I/O
    d. Interrupts use up little processor time but polling gives low latency

    Interrupts may take longer (since you have to go through the interrupt handler) but they allow the processor to do other things while it is waiting for I/O. Polling, however, will much more quickly let you know when data is available since you don't have to go through the interrupt handler, but you can't do anything else while you are polling.

2. What main problem does DMA solve?
`2013-Dec-Input/Output-2`
    a. How to know when the IO device is ready
    b. How to access the IO device
    c. Having to write a program to move the data
    d. All of the above

    DMA just provides an easy way to have something other than the processor handle the data transfer. It uses the standard methods for accessing the IO device (e.g., memory-mapped) and for knowing when it is ready (e.g., polling or interrupts).

3. What technologies are shown in this cost per MB vs. year plot?

2013-Dec-Input/Output-3

a. A=DRAM, B=Flash, C=Hard disk
b. A=Hard disk, B=Flash, C=DRAM
c. A=Flash, B=DRAM, C=Hard disk
d. A=Hard disk, B=DRAM, C=Flash

The cheapest one is hard disk (A), the most expensive is DRAM (C), and flash is in the middle.

4. Why are random accesses (e.g., read 7,12,3) so much slower than sequential ones (e.g., read 1,2,3) on a hard disk?
2013-Dec-Input/Output-4
   a. The controller overhead can dominate for fast transfers
   b. The time it takes to move to a different track is very high
   c. The outer tracks can be read faster than the inner tracks
   d. All of the above

   Random accesses need to jump between tracks almost all the time (the chance the data is on the same track is small) so they spend all of their time moving the head. Sequential ones access all the data in the same track an so they only need to move the head after every track.

5. Why is there such a large difference between the best and worst throughput for sequential hard disk accesses?
2013-Dec-Input/Output-5
   a. The controller overhead can dominate for fast transfers
   b. The time it takes to move to a different track is very high
   c. The outer tracks can be read faster than the inner tracks
   d. All of the above

   Sequential accesses on the outermost track are faster because the disk circumference is larger the further out you go, which means that more sectors (more data) go past the head for every revolution.

6. As I/O devices become faster and faster they are moving to serial communication instead of parallel busses. What is the main reason for this?
2013-Dec-Input/Output-6
   a. Serial is cheaper to build, so we can make processors cheaper
   b. Serial is simpler to design, so we can design processors more quickly
   c. Serial is easier to synchronize, so we can communicate faster
   d. Hard to fit the parallel wires in a design, so we need to use fewer

   It is hard to synchronize the signals on parallel busses because the wires are all slightly different.

7. Why have busses become a problem in modern systems?
2017-Jan-Input/Output-1
   a. Hard to make small wires
   b. Hard to connect many wires
   c. Hard to find space for long wires
   d. Hard to keep the data on the wires synchronized

   The wires all have slightly different resistance and capacitance so they run at different speeds and are hard to keep synchronized.

8. What is the benefit of DMA?
2017-Jan-Input/Output-2
   a. Faster to detect that data is ready than polling
   b. Easier to program than interrupts
   c. Processor can do other things at the same time
   d. All of the above

   DMA really just allows the processor to do other things while the transfer is going on.

9. What does Ethernet do to avoid corrupted data on its shared wires?
2017-Jan-Input/Output-3
   a. Re-send the data if a collision is detected
   b. Wait a random amount of time to try again

c. Increase the waiting time every time a collision is detected
d. All of the above

Ethernet uses a backoff strategy that increases the time between re-tries every time a collisions is detected.

10. Why do serial communications send two versions of the same signal?
`2017-Jan-Input/Output-4`
    a. To check against each other to make sure the final signal is correct
    b. To subtract from each other to cancel out noise that affects both signals
    c. To reduce energy by sharing noise between the wires
    d. All of the above

    The two signals allow the effect of common noise to be eliminated by subtracting the signals from each other.

11. What are the tradeoffs between interrupts and polling?
`2017-Jan-Input/Output-5`
    a. Interrupts use up little processor time but polling gives low latency
    b. Interrupts are good for low-latency I/O and polling for high-throughput I/O
    c. Interrupts use up a lot of processor time but have a low latency
    d. Polling uses up little processor time but has a high latency

    Interrupts may take longer (since you have to go through the interrupt handler) but they allow the processor to do other things while it is waiting for I/O. Polling, however, will let you respond much more quickly when data is available since you don't have to go through the interrupt handler, but you can't do anything else while you are waiting.

12. How do memory-mapped IO accesses and virtual memory interact?
`2017-Jan-Input/Output-6`
    a. VM can protect IO devices from programs that shouldn't access them
    b. VM can map physical IO device locations to any virtual address
    c. VM can keep certain devices out of a program's virtual address space to free up memory for it
    d. All of the above

    Virtual memory provides the same flexibility for IO devices as for any memory address, but many OSes don't take advantage of it to simplify things. E.g., windows maps 1GB of address space for the kernel to avoid having to do some mappings, which reduces the amount of space available to the application.

13. How can a solid state (flash) disk have both lower latency and higher throughput than a traditional (spinning) hard disk?
`2014-Apr-Input/Output-1`
    a. They have multiple read heads so they can read in parallel
    b. They have lower controller overhead so each read takes less time to start
    c. They have has no moving parts so it doesn't have to wait for the data to physically move to the right place to access
    d. All of the above

    Hard disks have multiple read heads and flash doesn't have a read head. The controller overhead isn't that much different (although they do very different things). The big issue is that flash isn't a spinning chunk of rust, so it is much faster to access the data.

14. Why is the throughput enormously lower for reading small chunks of random data from a spinning hard disk than for reading large chunks?
`2014-Apr-Input/Output-2`
    a. It takes a long time to move the read head between each small chunk
    b. The small chunks are stored on the inside track of the disk which is slower
    c. The controller has to process more to find many small chunks than one large chunk
    d. All of the above

    The overhead is due to having to move around for each chunk vs. moving once to the data and then just reading it as the disk moves by. The small chunks are stored randomly on the disk so they are not all in the slowest inside track. Processing many small chunks is not much harder than a large one since the large one is made up of many sectors internally.

15. Why are the wires on circuit boards often not straight?

2014-Apr-Input/Output-3

a. To try and fit more wires on the board
b. To make them the same speed as other wires
c. To save on copper
d. To reduce their capacitance and inductance

To make multiple wires the same effective speed the layout tries to give them similar RLC properties by adusting their length and placement.

16. Why does a serial link transmit both + and - versions of the signal?

2014-Apr-Input/Output-4

a. To save energy
b. To increase speed
c. To decrease noise
d. All of the above

By having both + and - versions the receiver can subtract them and cancel out the effect of noise that affected both wires the same way.

17. Why does polling give you the lowest latency for I/O?

2014-Apr-Input/Output-5

a. The transfer is handled by hardware so you don't have to wait for the CPU
b. There is no other code running so you can process the data quickly
c. The hardware interrupts the current code to tell you when the I/O is ready
d. All of the above

When you are polling your processor isn't doing anything else so you get to know that the data is ready very quickly. With an interrupt you have to wait for the OS and hardware to switch to the right part of the software. DMA hardware transfers don't tell you until after the transfer is finished that the data is ready.

18. Why does DMA give you the highest throughput?

2014-Apr-Input/Output-6

a. The transfer is handled by hardware so you don't have to wait for the CPU
b. There is no other code running so you can process the data quickly
c. The hardware interrupts the current code to tell you when the I/O is ready
d. All of the above

In DMA the hardware moves the data so you don't have to use a program on the CPU to do it. Polling doesn't help because the CPU would still have to copy the data. And an interrupt doesn't help because you still have to copy the data in the CPU.

19. Why have busses become a problem in modern systems?

2015-Oct-Input/Output-1

a. Hard to make small wires
b. Hard to connect many wires
c. Hard to find space for long wires
d. Hard to keep the data on the wires synchronized

The wires all have slightly different resistance and capacitance so they run at different speeds and are hard to keep synchronized.

20. What is the benefit of DMA?

2015-Oct-Input/Output-2

a. Faster to detect that data is ready than polling
b. Easier to program than interrupts
c. Processor can do other things at the same time
d. All of the above

DMA really just allows the processor to do other things while the transfer is going on.

21. How would you access a device if you wanted the lowest possible latency?

2015-Oct-Input/Output-3

a. Polling
b. Interrupt
c. I/O instructions
d. DMA

Polling is the lowest latency, but highest CPU usage, way to access. Because your processor is constantly checking you will see if data is ready soonest. I/O instructions are one way to do polling, but you could also use them in an interrupt. DMA is just a way to transfer data, not to find out if the data is ready.

22. Why do we not use Flash memory instead of DRAM?
`2015-Oct-Input/Output-4`
    a. Too expensive
    b. Too slow
    c. Doesn't keep data without power
    d. Can't store enough data

Flash memory is cheaper than DRAM and keeps its data with no power, but it is still much slower than DRAM.

23. How does Ethernet share a single serial connection?
`2015-Oct-Input/Output-5`
    a. Each device is assigned different times to transmitToo slow
    b. Each device checks for data corruption and re-tries if needed
    c. Uses encryption to keep data safe from corruption
    d. Actually has separate wire for each device

Ethernet looks to see if the checksum for each packet has been corrupted by another device trying to use the same wire and, if so, retries.

24. How can memory-mapped IO accesses and virtual memory interact?
`2015-Oct-Input/Output-6`
    a. Virtual memory allows more I/O devices per application than the ISA's address space
    b. Page protection can limit which programs can access I/O devices directly
    c. Copy-on-write is used to get new data each time you read from the I/O address
    d. All of the above

Page protection can prevent programs from directly accessing memory-mapped IO devices by preventing them from accessing the addresses that talk to those devices. Virtual memory doesn't allow more I/O devices than the ISA address space since you still need addresses in the address space to access each one. You get new data each time you read from an I/O address because the device provides new data, not because of a copy-on-write.

25. What is responsible for setting up the memory address mappings for I/O devices?
`2018-Oct-Input/Output-1`
    a. The Program
    b. The Operating System
    c. The TLB
    d. All of the above

The operating system is responsible for identifying all the I/O devices and assigning them memory mappings. If the program did that, it would have to have direct access to those devices, which would be a security nightmare. The TLB is what allows the program to quickly use page mappings, but does not set them up.

26. What are the benefit(s) of polling over interrupt-based I/O?
`2016-Aug-Input/Output-1`
    a. Polling can detect ready data sooner
    b. Polling uses less processor time
    c. Polling has higher throughput for large data transfers
    d. All of the above

Polling can detect that data is ready sooner because that's all the processor is doing. Once it detects the data is ready, the transfer is the same rate as interrupt-based I/O and the cost of doing this is more processor time spent checking if the data is ready.

27. What is the benefit of DMA for data transfers?
`2016-Aug-Input/Output-2`
    a. Uses regular memory instructions to move data
    b. Simplifies the hardware design
    c. Reduces the processor time for moving data
    d. Reduces the latency of checking if data is ready

DMA is a way of having special hardware do the data movement instead of having the processor have to do it. This reduces the processor time for moving the data, but the processor still has to check if the data is ready to be moved.

28. Which of these is not a benefit of memory-mapped I/O?
`2016-Aug-Input/Output-3`
    a. Lower-latency
    b. Uses regular instructions
    c. Provides the same type of access to all data
    d. Enables virtual memory to protect I/O devices

Memory-mapped I/O allows you to access all devices (and main memory) through regular load/store instructions, which allows virtual memory to apply the same protections it would to any memory access. However, it doesn't reduce the latency, but rather just makes it easier to access them.

29. A disk has a memory mapped address for sending it data and has worked fine. A developer just got a new processor with a cache, and finds that the data from the disk is being corrupted such that only the last value sent *to* the disk from the software is being written to disk because of the cache. What change could be made to fix this problem?
`2016-Aug-Input/Output-4`
    a. Use a write-through cache
    b. Use a write-back cache
    c. Use a fully-associative cache
    d. None of these will solve the problem

If you use a write-through cache, then all the data written will be sent out of the cache as well as being stored there. This will make sure that all writes get to the device.

30. A disk has a memory mapped address for sending it data and has worked fine. A developer just got a new processor with a cache, and finds that the data is being corrupted such that only the first value sent *from* the disk to the processor is being read by the software. What change could be made to fix this problem?
`2016-Aug-Input/Output-5`
    a. Use a write-through cache
    b. Use a write-back cache
    c. Use a fully-associative cache
    d. None of these will solve the problem

The problem is that the cache has the memory-mapped address for the read in the cache and just returns it each time the processor reads. It needs to go to the device on every read to make sure it gets the new data. The only way to fix this is to make sure that address is never cached.

31. How does differential signaling help in communications?
`2016-Aug-Input/Output-6`
    a. Increases speed by sending two bits at the same time
    b. Puts the clock into the signal by using two opposite wires
    c. Improves synchronization by requiring only two wires
    d. Reduces the effect of noise on the wires by subtracting it out in the end

The two wires in differential signaling both experience the same noise, and by subtracting one from the other at the end you can subtract out this noise. Since the signals are sent as opposites, this increases the signal at the end.

32. Which of the following statements are true about hard disks?
`2014-Dec-Input/Output-1`
    a. The worst-case for accessing the data is determined by the time it takes to move the head from the inner-most track to the outer-most track
    b. The worst-case for accessing the data is determined by the time it takes to spin the disk all the way round
    c. The worst-case for accessing the data is the max of A or B
    d. The worst-case for accessing the data is determined by the disk's memory controller, which has to queue all the

requests.

The worst-case is if you have to move the head all the way from the outside to the inside (or vice-verse) and wait for the disk to spin all the way round to get the data under the head. These two movements happen at the same time, though, so it is the max.

33. Which of the following is NOT a benefit of memory-mapped IO?
`2014-Dec-Input/Output-2`
    a. Faster
    b. Uses regular instructions to access IO
    c. Unifies all the device's memories
    d. Enables the use of virtual memory to protect IO devices

With memory-mapped IO you just need to know the address of the device and then do a regular load/store to access it. Memory-mapping is just a way to get the data, it does not change the performance per se. All the internal memories from the devices are unified since they are mapped to main memory address space.

34. Rank the following technologies in terms of speed: flash, DRAM, hard disk, tape, SRAM
`2014-Dec-Input/Output-3`
    a. (fastest) SRAM < DRAM < hard disk < tape < flash
    b. (fastest) SRAM < DRAM < flash < hard disk < tape
    c. (fastest) flash < SRAM < DRAM < hard disk < tape
    d. (fastest) DRAM < SRAM < flash < hard disk < tape

Fastest is SRAM (caches), which is followed by DRAM (Main memory). Flash memory is faster than hard disk but slower and cheaper than DRAM and that is why it is used in Solid State Drives. Finally hard disks and tapes are incredibly slow and insanely slow, respectively.

35. Why would a bank use polling to check for new network data instead of an interrupt?
`2014-Dec-Input/Output-4`
    a. Because DMA has too low a throughput
    b. Because they need to change their software so often that it is better to write a quick-and-dirty loop to check for data than to use interrupts
    c. Because they are willing to waste a processor polling to get lower latency so they can trade stocks faster
    d. All of the above

Bankers care about making trades faster than their competitors. (Most stock trading today happens via automated systems in only a few ms.) They need the lowest latency, and polling gets that by having the CPU do nothing but check for data.

36. Which performance change would help the most to make it take equally long to read two different 1024-byte sectors vs. one 2048-byte sector from a hard disk?
`2014-Dec-Input/Output-5`
    a. Decrease seek time
    b. Increase transfer rate
    c. Decrease rotation speed
    d. Decrease controller overhead

The problem is the time it takes to get to another location, which is a function of the seek time and the rotation speed. We need to decrease the seek time and/or increase the rotation speed. The controller overhead is tiny compared to the time it takes to move the disk head to the data.

37. A programmer writes a program to turn off a light by writing the OFF command to the memory-mapped address for the light controller. The programmer notices that the light does not turn off right when the program sends the command, but seems to happen only after the program accesses a lot of data after writing the OFF command. What is the likely explanation for this?
`2014-Dec-Input/Output-6`
    a. There is a write-back cache
    b. The DMA controller is not finished
    c. The program does not have write permission for the light controller's physical memory address
    d. Any of the above

A write-back cache will keep the write that would have gone to the light controller in the cache until it gets evicted.

This is consistent with the command not getting there right away, and accessing lots of memory (which would cause the command's cache line to be evicted) causing it to be updated. There is no DMA controller involved since we are writing to the memory-mapped address directly, and if it was a page permission problem, then the program would crash and it would never turn on.

38. Why would you choose to use interrupts instead of polling?
`2015-Apr-Input/Output-1`
    a. Polling has lower latency than interrupts
    b. Polling uses up less processor time
    c. Interrupts use up less processor time
    d. Interrupts have lower latency

Interrupts may take longer (since you have to go through the interrupt handler) but they allow the processor to do other things while it is waiting for I/O. Polling, however, will much more quickly let you know when data is available since you don't have to go through the interrupt handler, but you can't do anything else while you are polling.

39. Which of the following is true when it comes to making busses fast?
`2015-Apr-Input/Output-2`
    a. Wires interfere with their neighbors
    b. Wires of different lengths take different amounts of time
    c. To send more bits at the same time you need more wires in a bus
    d. All of the above

Busses need many wires working together to send bits in parallel, and these wires interfere with the ones next to them, and take different amounts of time to send data if they are different lengths.

40. Why does reading large chunks of contiguous data from a spinning hard disk give enormously higher throughput than reading small chunks of data from random locations?
`2015-Apr-Input/Output-3`
    a. The small chunks are stored on the inside track of the disk which is slower
    b. Moving the read head between each small chunk takes a long time
    c. The controller has to process more to find many small chunks than one large chunk
    d. All of the above

The overhead is due to having to move around for each chunk vs. moving once to the data and then just reading it as the disk moves by. The small chunks are stored randomly on the disk so they are not all in the slowest inside track. Processing many small chunks is not much harder than a large one since the large one is made up of many sectors internally.

41. What does Ethernet not do to avoid errors?
`2015-Apr-Input/Output-4`
    a. Detect data collisions
    b. Use separate wires for each sender
    c. Retry sending the data
    d. Use a checksum

Ethernet uses a checksum to detect if the data has been corrupted by collisions and then retires sending it later. It does not use separate wires for each sender, but relies on a shared medium.

42. How do DMA and memory-mapping interact?
`2015-Apr-Input/Output-5`
    a. DMA cannot be used if the device is memory-mapped
    b. DMA uses the device's interrupt to move data
    c. DMA moves data by accessing the device directly
    d. DMA moves data using the device's memory-mapped addresses

DMA just copies data using the same memory-mapped addresses as regular code might. The DMA avoids having to have a program running on the CPU take care of the copy.

43. How does DMA make our computers faster?
`2015-Apr-Input/Output-6`
    a. It transfers data faster than the CPU can

b. It avoids the overhead of interrupts when data is ready
c. It frees up the CPU from having to do the transfer
d. All of the above

DMA just relieves the CPU of the need to do the transfer itself so it can do other work.

44. Why don't we use DRAM for data storage instead of disks and flash?
`2013-Jan-Input/Output-1`
    a. Too expensive
    b. Too slow
    c. Too unreliable
    d. Something else

45. What is the problem with making Flash memory smaller and smaller in the future?
`2013-Jan-Input/Output-2`
    a. Too expensive
    b. Too slow
    c. Too unreliable
    d. Something else

46. What is the problem with busses for transmitting data off-chip?
`2013-Jan-Input/Output-3`
    a. Too many wires to fit on the board
    b. Wires are hard to synchronize
    c. Too hard to make the wires straight
    d. All of the above

47. What does differential signaling in serial communications solve?
`2013-Jan-Input/Output-4`
    a. Increases speed by sending two bits at the same time
    b. Improves synchronization by requiring only two wires
    c. Reduces noise effects by subtracting out the noise at the end
    d. Puts the clock into the signal by using two opposite wires

48. What is the benefit of DMA?
`2013-Jan-Input/Output-5`
    a. Uses the memory space for addressing I/O devices
    b. Relieves the processor of the burden of having to check if the transfer is done
    c. Simplifies the hardware design
    d. Relieves the processor of the burden of managing the data transfer

49. What does Ethernet do to avoid corrupted data on the shared wire?
`2013-Jan-Input/Output-6`
    a. Keep sending data if a collision is detected
    b. Wait a random amount of time and try again
    c. Increase the waiting time every time a collision is detected
    d. All of the above

50. What are the tradeoffs between interrupts and polling?
`2015-Aug-Input/Output-1`
    a. Interrupts use up little processor time but polling gives low latency
    b. Interrupts are good for low-latency I/O and polling for high-throughput I/O
    c. Interrupts use up a lot of processor time but have a low latency
    d. Polling uses up little processor time but has a high latency

Interrupts may take longer (since you have to go through the interrupt handler) but they allow the processor to do other things while it is waiting for I/O. Polling, however, will let you respond much more quickly when data is available since you don't have to go through the interrupt handler, but you can't do anything else while you are waiting.

51. What problem does DMA solve?
`2015-Aug-Input/Output-2`
    a. How to access an IO device

b. How to move data to/form an IO device
c. How to know when an IO device is ready
d. All of the above

DMA makes it easy to move data to/from a device by having the processor copy the data directly from memory.

52. What problem does memory-mapping solve?
`2015-Aug-Input/Output-3`
    a. How to access an IO device
    b. How to move data to/form an IO device
    c. How to know when an IO device is ready
    d. All of the above

Memory-mapping solves the problem of accessing a device by making it look like a regular memory access.

53. How does Ethernet ensure reliable communication?
`2015-Aug-Input/Output-4`
    a. Sending both + and - versions of the data
    b. Using a checksum to verify the data we receive
    c. Reading the data as we send it to see if it is corrupted
    d. All of the above

Ethernet is a differential serial protocol that shares wires with other devices. As a result it needs to detect if there is a collision. It does this through using a checksum to see if the received data was corrupted and reading its own data back as it transmits to see if another device is sending at the same time and corrupting its data.

54. Why is it hard to make busses fast?
`2015-Aug-Input/Output-5`
    a. Wires become slower as they get smaller
    b. Wires interfere with other wires
    c. Wires of different lengths take different times to send data
    d. All of the above

All of the above are true. Making busses faster is very hard, which is why we are seeing a shift toward serial links.

55. DRAM is faster than Flash (NVRAM) and Hard disks. So why do we not use it for permanent storage?
`2015-Aug-Input/Output-6`
    a. Too expensive
    b. Slower than SRAM
    c. Doesn't keep data without power
    d. We do use it for permanent storage

DRAM would be excellent storage until the power went out. At that point all the capacitors in the DRAM would eventually lose their charge and you would lose your data.

56. What is the benefit of DMA?
`2014-Sep-Input/Output-1`
    a. Faster than polling
    b. Easier than interrupts
    c. Processor can do other things at the same time
    d. All of the above

DMA really just allows the processor to do other things while the transfer is going on.

57. What is hard about building communications busses?
`2014-Sep-Input/Output-2`
    a. Using enough wires to get the bandwidth
    b. Synchronizing the signals on all the wires
    c. Finding space in servers
    d. All of the above

Fast busses are hard to build because you have to keep the signals synchronized between all the wires.

58. Why do serial communications send two versions of the same signal?
`2014-Sep-Input/Output-3`
    a. To check against each other to make sure the final signal is correct
    b. To subtract from each other to cancel out noise that affects both signals
    c. To reduce energy by sharing noise between the wires
    d. All of the above

    The two signals allow the effect of common noise to be eliminated by subtracting the signals from each other.

59. Ethernet networks are a serial link that is shared by multiple computers at the same time. How does Ethernet make sure that other computers do not send at the same time and corrupt the data?
`2014-Sep-Input/Output-4`
    a. The receiver checks the checksum to verify the data hasn't been corrupted
    b. The sender checks the sent data to see if it is what was expected
    c. The sender automatically retransmits if the data was corrupted
    d. All of the above

    Ethernet does all of the above and it works really well. Similar approaches are taken for many WiFi networks, although cellular networks are increasingly using centrally scheduled transmission plans wherein the base station decides who gets to send and when.

60. What are the benefits of hard disks over flash memory?
`2014-Sep-Input/Output-5`
    a. Faster
    b. Cheaper
    c. Smaller
    d. None

    Hard disks are still cheaper than flash, but they are much slower and larger.

61. How do memory-mapped IO accesses and virtual memory interact?
`2014-Sep-Input/Output-6`
    a. VM can protect IO devices from programs that shouldn't access them
    b. VM can map physical IO device locations to any virtual address
    c. VM can keep certain devices out of a program's virtual address space to free up memory for it
    d. All of the above

    Virtual memory provides the same flexibility for IO devices as for any memory address, but many OSes don't take advantage of it to simplify things. E.g., windows maps 1GB of address space for the kernel to avoid having to do some mappings, which reduces the amount of space available to the application.

62. Which of these applications is *not* latency-sensitive?
`2016-Oct-Input/Output-1`
    a. A video call application
    b. Online video games
    c. Downloading a very big file as fast as possible
    d. Loading your Facebook feed

    When we download a file, we don't care about the time it takes to get the first byte. We want the whole file as quickly as possible, which requires a high bandwidth.

63. Which of these applications would benefit the most from using polling to access its data?
`2016-Oct-Input/Output-2`
    a. A text editor (reads the file once at startup)
    b. A backup program (backing up a large number of large files)
    c. A chat client (responding to a message from a server)
    d. A high frequency trading software (responding to changes in the market)

    High frequency trading software need the lowest latency in order to take decisions as fast as possible. This can cause market crashes.

64. Why are solid-state (NVRAM/FLASH) drives so much faster than regular hard drives?

2016-Oct-Input/Output-3

a. Because they use no electronic components, only mechanical parts
b. Because they use no mechanical parts, only electronic components
c. Because they need to be constantly powered in order to retain data
d. Because they use SRAM cells, similar to CPU caches

Hard drives use a spinning disks and a mechanical head to access data. SSDs on the other hand use only electronic components. They do not need to be constantly powered (otherwise, you would lose all your data each time you turn off your computer). They typically use Flash memory, not SRAM.

65. What is the main bottleneck when reading many different 1KB chunks from a hard drive?

2016-Oct-Input/Output-4

a. Rotation time
b. Seek time
c. Data transfer
d. Controller overhead

The time to move the magnetic head to the right location is the biggest overhead.

66. How are interrupts and DMA used together?

2016-Oct-Input/Output-5

a. They are completely unrelated
b. The CPU can use an interrupt to interrupt the data transfer
c. The device interrupts the CPU when the DMA transfer is over
d. The DMA engine interrupts the CPU when the DMA transfers is over

Once a DMA operation completes, the DMA engine raises an interrupt to signal it is done.

67. Why would you use DMA?

2016-Oct-Input/Output-6

a. Using DMA, the I/O operation can be overlapped with other computation
b. Using DMA reduces the CPU overhead caused by the I/O operation
c. Using DMA, the I/O operation completes faster
d. A and B

When using DMA, once the transfer is initiated, the application can perform other tasks.
Moreover, once initiated, the transfer does not involve the CPU, thus reducing the CPU overhead of the I/O operation compared to programmed I/Os. It has however no impact on the time it takes to complete the transfer, since this is largely limited by the device itself

68. What is the benefit of transmitting both + and - versions of a signal?

2016-Jan-Input/Output-1

a. Faster transmission
b. Better noise immunity
c. Lower energy
d. All of the above

With both versions of the signal you can subtract them to eliminate common noise on the wires.

69. Which of the following is NOT a benefit of memory-mapped IO?

2016-Jan-Input/Output-2

a. Uses regular instructions
b. Faster access to data
c. Unifies all device memory spaces
d. Enables VM to provide device protection

Memory mapped IO is just a way to access the data and does not affect the intrinsic speed with which you can transfer it.

70. Why is it important that wires on a circuit board are the same effective length?

2016-Jan-Input/Output-3

a. So the signals arrive at the same time

b. So the signals don't affect other wires
c. So they pack densely on the board
d. To reduce their capacitance

Making the wires the same effective length means that the signals will all arrive at the end at the same time. Note that "effective" is used here because "length" isn't really what affects the transmission time: it is the capacitance and inductance, but those are both affected by the length and the layout.

71. How can solid state (flash) disks have both lower latency and higher throughput than a traditional (spinning) hard disk?
`2016-Jan-Input/Output-4`
    a. They do not have to wait for moving parts to physical move to the data
    b. They have multiple read heads so they can read in parallel
    c. They have lower controller overhead so reads take less time to start
    d. All of the above

    Hard disks have multiple read heads (one or two for each platter) but the reason they are slow is that they require the read heads to physically move to the data location. Flash only needs to send electrical signals to the read location, which is far, far faster.

72. Why would a programmer use interrupts for IO instead of polling?
`2016-Jan-Input/Output-5`
    a. To get lower latency
    b. To get higher throughput
    c. To get other work done while waiting
    d. All of the above

    Interrupts allow the program to continue working while waiting for the IO, but they do so at the cost of increased latency.

73. Which of the following is NOT a technique Ethernet uses to enable multiple computers to send and receive data?
`2016-Jan-Input/Output-6`
    a. Check if another computer corrupted the received data with checksums
    b. Check if another computer corrupted the sent data by reading it back while sending
    c. Use separate wires for each computer sending and receiving
    d. Wait a random amount of time to retry a corrupted transmission

    In Ethernet, the sender reads back the data it transmits to detect if another computer corrupted it while sending and sends a checksum so other computers can detect if it is corrupted as well. If there is a corrupted transmission, the computer will wait a random (and exponentially increasing) amount of time before retrying.

74. What is the worst-case for accessing data on a hard disk?
`2013-Aug-Input/Output-1`
    a. The time it takes to move the head from the inner-most track to the outer-most track
    b. The time it takes to rotate the disk all the way around
    c. The sum of A and B
    d. All data takes the same amount of time to access

    The worst-case is if you have to move the head all the way from the outside to the inside (or reverse) and then wait for the disk to spin all the way around to get the data under the head.

75. Rank the following technologies in terms of cost per MB: flash, DRAM, hard disk, tape
`2013-Aug-Input/Output-2`
    a. (cheapest) flash < DRAM < hard disk < tape
    b. (cheapest) tape < hard disk < DRAM < flash
    c. (cheapest) tape < DRAM < flash < hard disk
    d. (cheapest) tape < hard disk < flash < DRAM

    Tape is the absolute cheapest (just buy more tapes), hard disks are next cheapest, and then flash. DRAM is 10x more expensive than flash.

76. Why is it hard to make busses fast?
`2013-Aug-Input/Output-3`

a. Wires interfere with their neighbors
b. Wires of different lengths take different amounts of time
c. Wires get slower as they get smaller
d. All of the above

All of the above.

77. What is the benefit of differential serial signaling over busses?
`2013-Aug-Input/Output-4`
    a. Differential signals resist noise because it gets subtracted out at the end
    b. Differential signals resist noise because it gets added in at the end
    c. Serial signals are easier to build than parallel busses
    d. Serial signals use a slower clock because they send data in parallel

    With differential signals noise affects both wires similarly, and by subtracting them in the end, you can subtract out the noise.

78. What is the benefit of memory-mapped IO?
`2013-Aug-Input/Output-5`
    a. Faster
    b. Automatically moves data
    c. Uses regular instructions to access IO
    d. Does not pollute the address space

    With memory-mapped IO you just need to know the address of the device and then do a regular load/store to access it. Memory-mapping is just a way to get the data; it doesn't change the performance per se or make it easier to use. It does pollute the address space because you need to reserve addresses for the IO.

79. If you want to write a program with the absolute lowest possible latency to access an IO device, what approach would you use and why?
`2013-Aug-Input/Output-6`
    a. Interrupts: the processor can do other work while it is waiting
    b. Interrupts: the operating system will handle telling the program when the device is ready
    c. Polling: the program will detect when the device is ready because it can't do anything else
    d. DMA: the processor doesn't have to do the transfer so it will be faster

    In polling the processor does nothing but check if the IO device is ready. This means it will know as soon as possible when it is. With interrupts you have to wait for the interrupt handler to execute first.

80. Why are busses a problem?
`2013-Apr-Input/Output-1`
    a. Hard to make wires
    b. Hard to connect wires
    c. Hard to find space for wires
    d. Hard to keep the wires synchronized

    The wires all have slightly different resistance and capacitance so they run at different speeds and are hard to keep synchronized.

81. What is replacing busses, and why?
`2013-Apr-Input/Output-2`
    a. Serial links - because they can connect many devices
    b. Serial links - because they don't need to be synchronized
    c. Serial links - because they are simpler
    d. Optical links - because they are easier to build

    Serial links allow us to send data faster because we don't have to keep data synchronized across multiple wires. They are more complex and only allow a single device, though.

82. What is the benefit of DMA?
`2013-Apr-Input/Output-3`
    a. Improves the speed of data transfers

b. Reduces the workload of the processor
c. Simpler to use than polling
d. Lower overhead compared to interrupts

DMA allows the DMA engine to handle the work of moving individual chunks of data so the processor doesn't have to. It can then do other things.

83. What is the benefit of polling?
`2013-Apr-Input/Output-4`
    a. Simpler to program
    b. Lower overhead
    c. Higher throughput
    d. All of the above

Polling allows the programmer to directly check if the data is ready, but it makes it hard to do other things with the program.

84. What is the benefit of interrupt-driven I/O?
`2013-Apr-Input/Output-5`
    a. Simpler to program
    b. Lower overhead
    c. Higher throughput
    d. All of the above

With an interrupt, the processor only has to deal with the I/O when it is ready.

85. How does Ethernet share a single serial bus?
`2013-Apr-Input/Output-6`
    a. Has special times for each device
    b. Checks for data corruption and re-tries
    c. Uses encryption to keep data safe
    d. Really uses one wire per device

Ethernet looks to see if the checksum for each packet has been corrupted by another device trying to use the same wire and, if so, retries.

10. Caches

1. Do caches make memory seem faster?
`2013-Dec-Caches-1`
    a. Yes
    b. Most of the time
    c. Only for special data
    d. No

Caches make memory seem faster most of the time, but if the data isn't in the cache then it doesn't help.

2. Why would we build a set-associative cache?
`2013-Dec-Caches-2`
    a. It's simpler than direct mapped
    b. It's more flexible than fully-associative
    c. It has a smaller cache line size
    d. It has some associativity but isn't too complex

Set-associative caches are a compromise between fully-associative and direct mapped. They have some degree of associativity but don't need to do comparisons for every cache line.

3. A 32-bit program accesses the words at addresses 0, 16, 32, 48 over and over again. How many total cache bits would you need to hold all of this data for a fully-associative cache with A) 32 byte lines and B) 8 byte lines? (ignore the valid and dirty bits)
`2013-Dec-Caches-3`
    a. A=256 bits, B=128 bits

b. A=283 bits, B=157 bits
c. A=512 bits, B=256 bits
d. A=566 bits, B=372 bits

A 32-byte line cache will need two lines (e.g., 0 and 16 in one line and 32 and 48 in the second line) and have 27 bits of tag per line. An 8-byte line cache will need four lines and have 29 bits of tag per line. Two 32-byte lines is 64 bytes 27*2 tag bits=566 bits, and 4 8-byte lines is 32 bytes plus 4*29 tag bits = 372. Each one holds 4 words or 4*4=16 bytes of data.

4. Assume you have a 2-way set-associative LRU cache with 4 entries and a cache line size of 1 bytes. What will the hit (H) miss (M) pattern be for the following access pattern: 0 1 2 3 7 0 3 8 1?
`2013-Dec-Caches-4`
   a. M M M M M M H M M
   b. M M M M M H M M H
   c. M M M M M H H M M
   d. M M M M M H H M H

a) is fully-associative LRU, b) is direct-mapped, c) is 2-way set associative LRU, and d) is an ideal cache that only misses on data it hasn't seen before.

5. What is true about a write-back cache vs. a write-through cache?
`2013-Dec-Caches-5`
   a. Faster when reading data we've recently written
   b. Faster when writing data we've recently written
   c. Need to keep track of dirty lines
   d. All of the above

Write back caches write new data to the cache and only update the main memory when that data is evicted from the cache. This makes them faster for reads and writes if the data was recently used (e.g., is still in the cache) but you do have to keep track of which data is dirty.

6. On average, what fraction of instructions are loads or stores?
`2013-Dec-Caches-6`
   a. 0.33
   b. 0.5
   c. 1.0
   d. 1.33

Every third instruction is a load or store (0.33) but since we have to load an instruction, we need to do 1.33 memory accesses per instruction.

7. What data is evicted on a miss in a LRU cache?
`2017-Jan-Caches-1`
   a. The data that has been in the cache for the longest time
   b. The data that has most recently been installed in the cache
   c. The data that has not been used for the longest time
   d. Misses do not evict data

LRU=Least Recently Used.

8. Which of the following statements are true about caches?
`2017-Jan-Caches-2`
   a. Caches are smaller and faster than DRAM
   b. Caches increase the chance of reusing data
   c. Caches are cheaper than DRAM
   d. All of the above

Caches are faster memories than DRAM, so recently used data is accessed faster if we have them. On the other hand, cache memories are also more expensive than DRAM so that's why they are smaller. Caches do not increase the chance of reusing data: that is a property of the program.

9. A program is reading every word (4 bytes) consecutively of a very large (4GB) array. Which of the following cache

configurations can achieve a hit ratio of more than 50%? Assume LRU replacement policy?

2017-Jan-Caches-3

a. Cache size: 16 B. Block size 4 B, Direct mapped
b. Cache size: 8 MB. Block size 8 B. 2?way associative
c. Cache size: 2 KB. Block size 32 B. Fully associative
d. All of the above

None of these caches are large, so they will not be able to fit all the data. Since the data is being read sequentially, if we have a large enough cache block size, then we can reuse the same block repeatedly. A 32B block will be reused 8 times for sequential words, giving a hit ratio of 7/8=87.5%.

10. Which of the following assumptions is not required for caches to be effective?

2017-Jan-Caches-4

a. The amount of data being used at any one time is small
b. Recently used data is likely to be used again
c. Data is accessed in regular patterns
d. Data near recently used data is likely to be used soon

Data doesn't have to be in regular patterns since the cache can store any combination of data. For caches to work, the recently used data needs to be reused and it needs to be small enough to fit in the cache. The idea of a cache line also assumes that nearby data is likely to be used soon.

11. What data will be in a 4-entry, fully-associative, LRU cache with one word per line after the following memory accesses? 0, 1, 2, 3, 4, 5, 3, 2, 1?

2017-Jan-Caches-5

a. 1 3 4 5
b. 1 2 3 5
c. 1 2 4 5
d. 0 3 4 5

In a fully associative LRU cache the most recently used data is always resident in the cache.

12. What data will be in a 4-entry, direct-mapped cache with one byte per line after the following memory accesses? Address: 0, 1, 2, 3, 4, 5, 3, 2, 1, 2, 5

2017-Jan-Caches-6

a. 1, 2, 3, 4
b. 0, 2, 4, 5
c. 2, 3, 4, 5
d. 0, 1, 3, 4

We fill up to 0,1,2,3. Then we replace 0-->4, 1-->5, 3 hit, 2 hit, 5-->1, 2 hit, 1-->5

13. How do caches provide the illusion of a large and fast memory system?

2014-Apr-Caches-1

a. The make access to all data faster
b. They make access to the right data faster
c. They make access to recently used data faster
d. They make access to the least recently used data faster

They make access to the recently used data (the data that is still in the cache) faster

14. Which of the following assumptions is not required for caches to be effective?

2014-Apr-Caches-2

a. Recently used data is likely to be used again
b. The amount of data being used at any one time is small
c. Data near recently used data is likely to be used soon
d. Data is accessed in regular patterns

Data doesn't have to be accessed in regular patterns. For caches to work the recently used data needs to be reused and it needs to be small enough to fit in the cache. The idea of a cache line assumes that nearby data is also likely to be reused.

15. How many possible locations are there for storing the address 0100 0101 1110 0100 in a 64kB 4-way set-associative write back cache with 32 byte cache lines?
2014-Apr-Caches-3
    a. 1
    b. 4
    c. 16
    d. 2048

A 4-way set-associative cache can store any address in one of four ways.

16. Which has a lower average memory access time?
#1: 32kB L1 cache 1 cycle + 256kB L2 cache 4 cycles (80% hit in the L1, 20% in the L2)
#2: 64kB L1 cache 2 cycles + 128kB L2 cache 3 cycles (90% hit in the L1, 10% in the L2)
2014-Apr-Caches-4
    a. #1
    b. #2
    c. Same
    d. Need to know something about the program

#1 AMAT: .8*1+.2*4=1.6, #2 AMAT: .9*2+.1*3 = 2.1

17. What percentage of the bits used for data in a 32kB (32,768 byte) direct-mapped write-back cache with a 64 byte cache line?
2014-Apr-Caches-5
    a. 512/528
    b. 64/528
    c. 512/16
    d. 64/512

32 bit address bits, 64 byte line means we have 6 bits for the word address in the line that aren't in the tag, 32,768 bytes in the cache at 64 byte lines is 512 total lines, which means we have 12 bits of address for the cache index, write back means we need a dirty bit, and we always need a valid bit. So each line has 64*8=512 data bits, 32-6-12=14 tag bits, and 2 flag bits: data/total bits = 512/(512+14+2)=512/528

18. How did we use caches to solve the problem with an average of 1.33 memory accesses per instruction?
2014-Apr-Caches-6
    a. Caches are faster, so it isn't a problem
    b. Caches store data for recent instructions so we don't need as many accesses
    c. We can have separate instruction and data caches to read both at the same time
    d. We have multiple levels of caches to make the cache seem larger

By having separate instructions and data caches (as virtually all processors today do) we can effectively have the separate data and instruction memories as we have shown in our pipeline diagrams.

19. A program is reading every word (4 bytes) consecutively of a very large (4GB) array. Which of the following cache configurations can achieve a hit ratio of more than 50%? Assume LRU replacement policy.
2015-Oct-Caches-1
    a. Cache size: 16 B. Block size 4 B, Direct mapped
    b. Cache size: 8 MB. Block size 8 B. 2-way associative
    c. Cache size: 2 KB. Block size 32 B. Fully associative
    d. All of the above

None of these caches are large, so they will not be able to fit all the data. Since the data is being read sequentially, if we have a large enough cache block size, then we can reuse the same block repeatedly. A 32B block will be reused 8 times for sequential words, giving a hit ratio of 7/8=87.5%.

20. A program is reading a 4x4 word matrix in "the wrong way", e.g., addresses 0, 4*4, 8*4, ...,. What is the lowest associativity that will have no conflict misses for a 32B LRU cache with a block size of 8 B?
2015-Oct-Caches-2
    a. 2-way
    b. 4-way
    c. It is impossible to fully avoid conflict misses with this access pattern
    d. There are no conflict misses with this memory pattern

A full matrix column can now fill up the cache exactly.

21. What data is evicted on a miss in a LRU cache?
`2015-Oct-Caches-3`
    a. The data that has been in the cache for the longest time
    b. The data that has most recently been installed in the cache
    c. The data that has not been used for the longest time
    d. Misses do not evict data

LRU=Least Recently Used.

22. Which of the following statements are NOT true about caches?
`2015-Oct-Caches-4`
    a. Caches are faster than the DRAM used for main memory
    b. Large cache lines improves spatial locality
    c. High associativity decreases the number of misses when you first access data
    d. LRU is not always the best replacement policy

Increasing associativity does not reduce the number of misses when you first access data because the data will still not be in the cache (first time). A larger cache block might help by pulling in other data for you.

23. What data will be in a 4-entry, fully-associative, LRU cache with one word per line after the following memory accesses: 0, 1, 2, 3, 4, 5, 3, 2, 1?
`2015-Oct-Caches-5`
    a. 1 3 4 5
    b. 1 2 3 5
    c. 1 2 4 5
    d. 0 3 4 5

In a fully associative LRU cache the most recently used data is always resident in the cache.

24. How many sets does a 4kB, 4-way set associative cache with an 8-byte line size have?
`2015-Oct-Caches-6`
    a. 4
    b. 64
    c. 128
    d. 512

4096 B /8 byte lines /4 ways per set = 128 sets.

25. A set-associative, write-back cache with a LRU replacement policy, sees the following accesses from the processor: lw A (miss), lw B (miss), lw A (hit), sw A (hit). What is changed in the cache on the last access?
`2018-Oct-Caches-1`
    a. The LRU order of the sets, to update the most recently used one
    b. The dirty bit, to keep track of whether the data needs to be written back
    c. Which address bits are used to index the cache
    d. The vlid bit, to indicate that the cache entry has data

The cache will have both A and B in it, with A being marked as the most recently used by the last lw A instruction. When the final, sw A, instruciton is executed, it will not change the LRU order (since A is already most recently used), not change the valid bit (since A is already valid), but it will set the dirty bit, because this is the first write to A. The address bits that are used to index the cache are set when the cache is designed, and don't chagne as the processor executes.

26. How many lines will each way store for a 2-way set associative 64kB cache with 32B lines?
`2018-Oct-Caches-2`
    a. 2048
    b. 1024
    c. 512
    d. 64

In total there will be 64kB/32B=2048 lines. If it is 2-way, that means each way will have 1024 lines.

27. What is the problem caches try to solve?
2016-Aug-Caches-1
    a. The ISA promised a full 32-bit address space
    b. We want a large and fast memory
    c. We want a cheap and reliable memory
    d. All of the above

Caches try to make all of the memory look fast: that is one single large and fast memory.

28. What is required of the data a program uses for caches to be effective?
2016-Aug-Caches-2
    a. Access a small amount of data at any given time
    b. Access the same data repeatedly
    c. Access data that is close together
    d. All of the above

Programs need to access only a small amount of data at a given time since caches are small. The data needs to be accessed repeatedly to get any benefit from the cache. (The first access takes the full DRAM time.) And the data needs to be close together so that each cache line doesn't waste lots of space.

29. Why does a direct-mapped cache not need to keep track of LRU (least recently used) information to figure out which data to replace when new data is loaded?
2016-Aug-Caches-3
    a. The data can be placed in any available set
    b. The data can only be placed in one location
    c. The dirty bit tells which data can be replaced
    d. It does need to keep track of LRU information

In a direct-mapped cache there is only one place for each address, so new data that is brought in can only replace the data in the one location it's address maps to. Associative caches have multiple locations for each address, so they need to keep track of LRU information.

30. What data will be in a 4-entry, 2-way set-associative, write-back, LRU cache with a one byte line after the following memory accesses?
1, 5, 0, 2, 1, 3, 6, 4, 2
2016-Aug-Caches-4
    a. 0, 2, 4, 6
    b. 1, 2, 3, 4
    c. 1, 6, 4, 2
    d. 2, 3, 4, 6

With two sets, one set will hold odd addresses and the other even. Since it is LRU, this means that the last two odd addresses and the last two even addresses will be in the cache, or 2, 4, 3, 1 = 1, 2,3,4.

31. What would be the difference in performance for this program between the following two caches: 1kB 64-byte line, write-through, direct-mapped and 32kB 64-byte line, write-back, 4-way set-associative? (Assume only the data array accessed in the loop may be cached and that the sum variable is in a register.)

```
int data[8192]; // each int is a word
int sum = 0;
for (int i=0; i<1024; i++)
  sum+=data[i];
```

2016-Aug-Caches-5
    a. The 1kB cache will be faster
    b. The 32kB cache will be faster
    c. The speed will be the same
    d. It will depend on the values in data

This code does not reuse the data, so whether it is in the cache or not will have no impact on performance. Both caches will be the same.

32. What is the benefit of having separate instruction and data caches?

2016-Aug-Caches-6

a. The processor can load an instruction at the same time as it loads/stores data
b. Instructions and data use separate addresses so they can't share a cache
c. The instruction cache never needs to have information written into it so it can be simpler
d. Instructions always access a full cache line so they are more efficient to store separately

With separate caches a ld instruction in the MEM stage can access the data cache at the same time as the IF stage fetches an instruction from the instruction cache. Both instructions and data share the same memory space so they can be in the same cache, and instructions will not use all the information in a full cache line if there are branches to other instructions, for example.

33. Which of the following statements is not true about caches?

2014-Dec-Caches-1

a. Caches do not have to be as big as the data the program needs to access
b. It takes longer to access a cache if it is full
c. Cache memories are faster than memories used for DRAM
d. Caches only help if the application reuses its data

Caches are SRAM usually, which is much faster than DRAM. If an application is not reusing data over time, then it does not matter if it is in the cache, since it will be bringing the data from DRAM anyways. Performance of caches does not decrease if they are full.

34. Intel's Ivy bridge processors have a 256kB 16-way set associative L2 cache with a 64-byte cache line size. How many cache lines will each way have?

2014-Dec-Caches-2

a. 2048
b. 512
c. 256
d. 64

256kB/64bytes = 4096 cache lines in total. The cache is 16-way, which means that each way will have 4096/16 = 256 cache lines.

35. Choose the most important reason write-back caches are better or worse than write-through caches:

2014-Dec-Caches-3

a. Write-back caches are better than write-through caches because they do not have to keep track of which data has been changed with a dirty bit, making them simpler.
b. Write-back caches are better than write-through caches because they do not have to write back the data to DRAM when it is changed, making them faster.
c. Write-through caches are better than write-back caches because they write data back to DRAM, making the data consistent all the time.
d. Write-through caches are better than write-back caches because they do not use a valid bit to keep the changed data, making the hardware smaller and energy efficient.

Write-through caches use a valid bit (D is incorrect). Write back caches are better than write-through (C incorrect) for performance. Write back caches are more complicated (which is why they need a dirty bit) and not any smaller (they need a valid bit to indicate whether the line has garbage or not). They are faster, though, because they do not have to write to memory every time.

36. Why is Average Memory Access Time (AMAR) a more useful value than the cache Miss Ratio (MR)?

2014-Dec-Caches-4

a. If AMAT goes up then the program runs more slowly, but a higher MR does not suggest that the program will be slower.
b. MR does not tell you how much missing in the cache slows you down.
c. AMAT takes into account the effect of every cache in the system while MR is only for one cache.
d. All of the above.

A higher MR almost always means a slower program. (For this class it always means that; for more advanced out-of-order processors with data prefetchers it may not mean that.) AMAT can take into account more than one cache, but MR can also do that. The real difference is that AMAT tells you how much the cache misses hurt (slow down) while MR only tells you what percentage of the time the cache isn't helping.

37. Accessing main memory takes 100 cycles and accessing a cache takes 1 cycle. What would the performance difference be

for this program on a machine with a write-back vs. write-through cache? (Assume the cache line size is 1 word, the cache is direct-mapped 32kB, and empty at the start of the program.)

```
for (int i=0; i<4,000; i++) {
        data[i] = data[i] + data[i+1];
}
```

2014-Dec-Caches-5

a. Same speed
b. Write-back cache is 1.5x slower
c. Write-back cache is 2x slower
d. Write-back cache is 3x slower

Every iteration does 2 loads and 1 write: Iteration 1 load(0)=miss, load(1)=miss, write(0); Iteration 2: load(1)=hit, load(2)=miss, write(1); etc. So in each iteration we will have one load miss and one write. If we have a write-back cache, this will be 1 cycle to load the hit, 1 cycle to do the write, and 100 cycles for the miss, or 102 cycles. If we have a write-through cache it will be 1 cycle to load the hit, 100 cycles to do the write and 100 cycles for the miss, or 201 cycles. Because the data fits in the cache and it was empty, we won't have to wait to evict any data when we bring in the new data.

38. A designer makes a mistake and builds a cache that has a MRU (most recently used) replacement policy instead of an LRU replacement policy. What is left in a 4-entry, fully-associative, MRU cache with 1 byte lines after the following address accesses: 1, 2, 3, 4, 5, 3, 5, 1, 5, 2

2014-Dec-Caches-6

a. 2
b. 2, 5
c. 2, 5, 3
d. 2, 5, 1, 3

The MRU policy means we replace whatever we last put in the cache. So we will only ever use one entry in the cache!

39. Why do we want caches?

2015-Apr-Caches-1

a. DRAM is slow
b. Disks are slower than DRAM
c. Virtual memory requires it
d. All of the above

We use caches because accessing data in DRAM is too slow. We use DRAM to cache data from disks, and virtual memory does not require a cache to operate.

40. Which of the following assumptions is required for caches to be effective?

2015-Apr-Caches-2

a. Recently used data is not likely to be used again soon
b. The amount of data being used at any one time is big
c. Recently used data is likely to be used again soon
d. Data is accessed in regular patterns

Data doesn't have to be accessed in regular patterns. For caches to work the recently used data needs to be reused and it needs to be small enough to fit in the cache so it will still be there when it is reused.

41. What data will be in a 4-entry, direct-mapped, cache with a least-recently-used replacement policy and one byte per line after the following memory accesses??address: 3, 2, 1, 0, 4, 5, 3, 1, 2

2015-Apr-Caches-3

a. 1, 2, 3, 5
b. 1, 2, 3, 4
c. 1, 3, 4, 5
d. 1, 2, 4, 5

We fill up to 3,2,1,0, then we replace 0-->4, 1-->5, 3 hit, 5-->1, 2 hit

42. What data will be in a 4-entry, fully-associative, cache with a least-recently-used replacement policy and one byte per line

after the following memory accesses??address: 3, 2, 1, 0, 4, 5, 3, 1, 2

2015-Apr-Caches-4

a. 1, 2, 3, 5
b. 1, 2, 3, 4
c. 1, 3, 4, 5
d. 1, 2, 4, 5

The fully-associative cache can keep any 4 values, and the LRU replacement policy will replace the oldest one. This will result in the last 4 distinct values being in the cache, which are 5,3,1 and 2.

43. Caches give programmers the illusion of what?

2015-Apr-Caches-5

a. Atomic load/store instructions in a pipeline
b. A full 32-bit address space for each application
c. A large and fast memory
d. All of the above

The cache provides the illusion of making memory look both fast and large by storing the right data in a small, fast memory.

44. What is the difference between a direct-mapped cache with a least-recently-used (LRU) replacement policy and a direct-mapped cache with a most-recently-used (MRU) replacement policy?

2015-Apr-Caches-6

a. The MRU policy will work less well, leading to a lower hit ratio
b. The MRU policy will work less well, leading to a higher hit ratio
c. The MRU policy will effectively have only one entry that gets used
d. The caches will behave the same

Direct-mapped caches only have one place to put every address, so there is no choice in terms of what you evict (or remove) when you install new data. Even using an incredibly stupid replacement policy like MRU will not have any impact. (For a fully-associative cache, an MRU policy would mean you always replace the last item you inserted, so you would only use one location.)

45. What is the problem that caches try to solve?

2013-Jan-Caches-1

a. ISA promised a full 32-bit address space
b. We want a large and fast memory
c. We want a cheap and reliable memory
d. All of the above

46. Caches work by doing what?

2013-Jan-Caches-2

a. Moving the current data into faster memories
b. Combining slower Flash and faster DRAM
c. Combining slower hard disk and faster DRAM
d. Making SRAM as cheap as DRAM

47. An N-line, direct-mapped, write-through cache needs what in addition to the data?

2013-Jan-Caches-3

a. N valid bits, N dirty bits, N tags, N comparators
b. N valid bits, N tags, N comparators
c. N valid bits, N dirty bits, N tags, 1 comparator
d. N valid bits, N tags, 1 comparator

48. A processor with a nominal CPI of 1.0 has one cache for all memory accesses. It takes 1 cycle for a hit and a miss takes 100 cycles. If 33% of the instructions are memory accesses and the cache has a miss ratio of 10%, what is the average memory access time per instruction?

2013-Jan-Caches-4

a. 3.3
b. 3.6
c. 13.3
d. 14.5

49. Which addresses are present in a LRU, 2-way, set associative, write-through, cache with 1 byte per line and 4 entries after the following addresses are accessed in this order?
    1, 2, 3, 4, 5, 2, 6
    2013-Jan-Caches-5
    a. 2, 4, 5, 6
    b. 2, 3, 5, 6
    c. 3, 4, 5, 6
    d. 1, 4, 5, 6

50. What percentage of the data in a cache with a 64-byte line is used by the following code?

```
int data[100]; // each int is a word
for (int i=0; i<96; i+=4;) {
  data[i]++;
  data[i+3]--;
}
```

    2013-Jan-Caches-6
    a. 100%
    b. 75%
    c. 50%
    d. 25%

51. What is the problem caches try to solve?
    2015-Aug-Caches-1
    a. We want a large and fast memory
    b. The ISA promised a full 32-bit address space
    c. We need to do a translation at the same time as accessing data
    d. All of the above

    Caches make most accesses faster, so most of the time it seems as though our memory is very fast. But we can still use the full (large) DRAM, it's just that when the data isn't in the cache it is much slower.

52. Which of the following assumptions is not required for caches to be effective?
    2015-Aug-Caches-2
    a. The amount of data being used at any one time is small
    b. Recently used data is likely to be used again
    c. Data is accessed in regular patterns
    d. Data near recently used data is likely to be used soon

    Data doesn't have to be in regular patterns since the cache can store any combination of data. For caches to work, the recently used data needs to be reused and it needs to be small enough to fit in the cache. The idea of a cache line also assumes that nearby data is likely to be used soon.

53. Which is not a benefit of making the cache line twice as large?
    2015-Aug-Caches-3
    a. Can store twice as much data in the cache
    b. Half as many address tags needed for the same capacity
    c. Nearby data is more likely to be in the cache
    d. Tags are smaller

    If you make the cache line twice as large you change the chunk in which you store the data, but not how much data you can store. E.g., each tag now keeps track of twice as much data. This means you need half as many tags and since each chunk of data is twice as large, nearby data is more likely to be in the cache. It also means that the tags are smaller, but only one bit smaller.

54. What is a downside of making the cache line twice as large?
    2015-Aug-Caches-4
    a. Programs that access data sequentially will waste space
    b. Programs that access data randomly will waste space
    c. Tag comparisons are slower
    d. The percentage of bits spent on tag storage is higher

Since the cache line is twice as large, every time you access a random address it will bring in twice as much data to the cache (the line). This means that, unless other random addresses are in the same line, you will only be able to have half as many random addresses in the cache since there are half as many lines.

55. Approximately how long does it take to load data from DRAM on a modern computer?
`2015-Aug-Caches-5`
    a. 1 cycle
    b. 10 cycles
    c. 100 cycles
    d. 1000 cycles

Intel has it down to about 70 cycles, but many systems take up to 200. GPUs take about 500 cycles. This means you can do 100 instructions for every piece of data you need to load from DRAM. Of course modern processors can typically do 4 instructions each cycle, so that's really 400 instructions for each cache miss.

56. How many possible locations are there for storing the address 0100 0101 1110 0100 in a 64kB fully-associative write back cache with 4096 byte cache lines?
`2015-Aug-Caches-6`
    a. 1
    b. 4
    c. 16
    d. 2048

A fully-associative cache can store any address in any location. A 64kB cache with 4096 byte cache lines will have 64kB/4kB=16 lines total, so there are 16 places you could store any address.

57. Which of the following statements are true about caches?
`2014-Sep-Caches-1`
    a. Caches are smaller and faster than DRAM
    b. Caches increase the chance of reusing data
    c. Caches are cheaper than DRAM
    d. All of the above

Caches are faster memories than DRAM, so recently used data is accessed faster if we have them. On the other hand, cache memories are also more expensive than DRAM so that's why they are smaller. Caches do not increase the chance of reusing data: that is a property of the program.

58. What data is stored in a cache with a most recently used replacement policy?
`2014-Sep-Caches-2`
    a. The most important data
    b. The most recently used data
    c. The least recently used data
    d. The data that is difficult to get from memory

An MRU cache throws out the most recently used data, so it keeps the oldest data alive. This would be a very foolish design.

59. How many lines will each way have in a 256kB, 4-way set associative cache with a 32-byte line size?
`2014-Sep-Caches-3`
    a. 8192
    b. 2048
    c. 1024
    d. 512

256kB / 32 bytes = 8192 lines total. The cache is 4-way associative, which means that each way will have 8192 / 4 = 2048 lines

60. What data will be in a 4-entry, direct-mapped cache with one byte per line after the following memory accesses?
Address: 0, 1, 2, 3, 4, 5, 3, 2, 1, 2, 5
`2014-Sep-Caches-4`
    a. 1, 2, 3, 4
    b. 0, 2, 4, 5
    c. 2, 3, 4, 5

d. 0, 1, 3, 4

We fill up to 0,1,2,3. Then we replace 0-->4, 1-->5, 3 hit, 2 hit, 5-->1, 2 hit, 1-->5

61. What percentage of the data in the array "data" will be reused at least once from a 32kB fully-associative cache with a 64-byte line?

```
int data[1024];                  // each int is a word
for  (int i=0; i<1022 ; i ++) {
        data[i] += data[i+1];
        data[i+1] -= data[i]
}
```

2014-Sep-Caches-5

a. 100%
b. 75%
c. 50%
d. 25%

Each line is 64 bytes and the loop touches every byte in the array at least twice, so 100% of the data in the array will see at least one reuse.

62. Calculate the cache miss ratio for this code (number of hits/number of cache accesses) assuming a 32kB fully associative cache with a 64-byte line (i.e. all the data fits in the cache and we don't have replacements).

```
int data[1024];                  // each int is a word
for  (int i=0; i<1022 ; i ++) {
        data[i] += data[i+1];
        data[i+1] -= data[i]
}
```

2014-Sep-Caches-6

a. Around 8%
b. Around 6%
c. Around 4%
d. 0%

The loop accesses data in the following pattern: address 0, 1, 1, 0 for each loop. The 0 will be a miss, but then the cache line will bring in the next 16 words, so accesses to 0-15 after that will be hits. That means that we have 1 miss and then 3 hits for the first loop iteration, followed by 6 more iterations with 4 hits each, or 1 miss for 7 iterations: 1 miss/(7 iterations*4 accesses per iteration) = 3.6% miss ratio

63. What data is stored in a cache with a least recently used replacement policy?

2016-Oct-Caches-1

a. The least recently used data
b. The most recently used data
c. The least frequently used data
d. The most frequently used data

An LRU cache throws out the least recently used data, so it keeps the most recent data alive.

64. Intel's Skylake processors have a 32KB 8-way set-associative L1 cache with a 64 bytes cache line size. How many cache lines will each way have?

2016-Oct-Caches-2

a. 8
b. 32
c. 64
d. 512

The cache has 32KB / 64 bytes = 512 cache lines. Since there are 8 ways, each way will hold 512/8 = 64 cache lines

65. Which of the following statement is not true?

2016-Oct-Caches-3

a. Increasing the associativity can help reduce the miss ratio

b. Increasing the associativity reduces the total cache capacity
c. It is not practical to make a large cache fully-associative
d. Associativity is a trade-off between performance, power consumption and chip area.

Increasing the associativity can reduce the miss ratio by reducing the number of conflicts. For each cache access, all the tags within a set must be read, leading to an increase in power consumption and complexity. Moreover, making a cache fully-associative is only practical for small cache, since all the tags in a fully associative cache needs to be compared to the tag field in the address.
Increasing the associativity has however no impact on the total cache capacity.

66. Intel's Skylake processors have a 256KB, 16-way set-associative L2 cache with a 64 bytes cache line size. Assuming 32-bits addresses, how are the bits in the address used to access the cache?

2016-Oct-Caches-4

a. 26 bits for the tag, 6 bits to choose a byte within a cache line
b. 6 bits for the tag, 18 bits for the index, 8 bits to choose a byte within a cache line
c. 8 bits for the tag, 18 bits for the index, 6 bits to choose a byte within a cache line
d. 18 bits for the tag, 8 bits for the index, 6 bits to choose a byte within a cache line

There are 64 bytes per cache line. Therefore we need 6 bits in order to choose a byte within a cache line (because 2^6 = 64). Moreover, the cache is divided in 256KB / 64B / 16 = 256 sets, so we need 8 bits to index the cache. We use the remaining 18 bits for the tag.

67. In the following code, what is the miss ratio of the first execution (I=0) of the inner-loop on a 8KB fully-associative LRU cache with a 64 bytes cache line size?

```
int array[4096] //each int is a word
for(int i = 0; i < 2; ++i)
   for (int j = 0; j < 4096; ++j)
      Array[j] = array[j] + 2;  // one read, one write
```

2016-Oct-Caches-5

a. Around 100%
b. Around 6%
c. Around 3%
d. Around 0%

A cache line can fit 16 words (integers). The first time a new cache line is accessed, a miss occurs since we need to bring the cache line in. However, the subsequent 31 memory accesses all touch the same cache line (Note how each iteration of the inner loop accesses 2 times the same element). Therefore, we get one miss every 32 memory accesses. The miss ratio is thus 1/32 ~= 3.1%

68. In the following code, what is the miss ratio of the second execution (I=1) of the inner-loop on a 8KB fully-associative LRU cache with a 64 bytes cache line size?

```
int array[4096] //each int is a word
for(int i = 0; i < 2; ++i)
   for (int j = 0; j < 4096; ++j)
      Array[j] = array[j] + 2;  // one read, one write
```

2016-Oct-Caches-6

a. Around 100%
b. Around 6%
c. Around 3%
d. Around 0%

Since we are looping through an array bigger than the cache size, and our cache uses the LRU policy, by the time we start the second execution of the inner-loop, the cache will contain only the second half of the array (the most recently used data). Therefore, during the second execution of the loop, the application will experience the same number of misses. This is a notorious shortcoming of the LRU policy: It is very ineffective when looping through a working set bigger than the cache. - note that since the answer here is the same as the previous one, we should accept answers here that were the same as the previous even if they are wrong.

69. Which property do caches not rely on to work well?

2016-Jan-Caches-1

a. Recently used data will be reused soon
b. Nearby data will be reused soon
c. The amount of data used at any one time is small
d. Data is accessed in a regular pattern

Caches don't care about the pattern of data access as long as the program reuses a small amount of data in a short time.

70. Why are caches so important?
2016-Jan-Caches-2
a. Processors are becoming faster more rapidly than memory
b. Memory is becoming faster more rapidly than processors
c. Processors need more memory
d. Programs need to access more data

Because processors are becoming faster more rapidly than memory we need caches to supply data quickly or the processors will stall.

71. A program reads a very large (2GB) array, but accesses only every 16th 32-bit word. How would changing the cache block size of a small (32kB) cache from 32B to 64B affect the hit ratio?
2016-Jan-Caches-3
a. More hits
b. Fewer hits
c. No change
d. Need more information

16 words is 64 bytes, which means a 64B cache block will only have on piece of used data per cache block. This is the same as a 32B cache block, so there will be no increase in hits due to the large cache block size. If the cache block was increased to 128B it would increase it.

72. A program reads data sequentially from a 2GB array over and over again. What would the performance impact be of doubling the cache size from 1MB to 2MB be if both caches take the same amount of time to access?
2016-Jan-Caches-4
a. Faster
b. Slower
c. No change
d. Need more information

Both caches will have the same hit ratio as changing the cache size from 1MB to 2MB won't increase the reuse since you would need a 2GB cache for that.

73. What data will be in a 4-entry, direct-mapped, LRU cache with one word per line after the following words are accessed?
0, 1, 2, 7, 8, 4, 3, 1, 5, 6
2016-Jan-Caches-5
a. 1 3 5 6
b. 3 4 5 6
c. 0 1 3 6
d. 1 4 5 6

Direct-mapped cache will have 4 in the 0-slot, 5 in the 1-slot (replacing 1), 6 in the 2-slot, and 3 in the 3-slot.

74. What is the difference between a fully-associative LRU (least recently used) and fully-associative MRU (most recently used) cache?
2016-Jan-Caches-6
a. The LRU policy will work less well, leading to a lower hit ratio
b. The MRU policy will work better, leading to a higher hit ratio
c. The MRU policy will effectively have only one entry that gets used
d. The caches will behave the same

A fully-associative cache with an MRU policy will always replace the last entry used in the cache, which will mean it will only ever have one entry with any data.

75. Why do we want caches?
2013-Aug-Caches-1
    a. DRAM is slow
    b. Processors are fast
    c. Lots of instructions access memory
    d. All of the above

Slow DRAM + fast processors wouldn't be a problem if we rarely accessed memory, but since 1/3 of instructions are loads or stores and all instructions need to load the instructions, we need to access memory very fast.

76. Approximately how many processor cycles does it take to access data in DRAM? (E.g., on a cache miss.)
2013-Aug-Caches-2
    a. 1
    b. 10
    c. 100
    d. 1000

About 100. E.g., you can do 100 instructions in the time it takes to access one piece of data from DRAM.

77. How many lines will each way have in 32kB 8-way set associative cache with a 64 byte line size?
2013-Aug-Caches-3
    a. 4096
    b. 512
    c. 64
    d. 8

32kB/64 bytes = 512 lines total. The cache is 8-way, which means that each way will have 512/8=64 lines,

78. Why are write-back caches better than write-through?
2013-Aug-Caches-4
    a. Simpler: use a dirty bit to keep track of which data has been changed
    b. Faster: don't have to write back data to DRAM when it is changed
    c. Smaller: don't need a valid bit because they keep the changed data
    d. All of the above

Write-back caches are more complicated (which is why they need a dirty bit) and not any smaller (they need a valid bit to indicate whether the line has garbage or not). They are faster, though, because you don't have to write to DRAM on every write.

79. How many hits will the following access pattern generate in a 2-way set associative LRU cache with 8 entries. (Assume the line size is 1 entry per line.) 4, 5, 6, 12, 13, 4, 5, 1, 12, 13, 5
2013-Aug-Caches-5
    a. 2
    b. 3
    c. 4
    d. 5

4, 5, 6 -> will put 4, 5, 6 in the first way. 12, 13 will put 12, 13 in the second way. These are all misses because the data isn't in the cache. 4, 5 are now hits. 1 is a miss, and replaces 13 because it is least recently used. 12 now hits, but 13 misses (it was replaced by 1) and replaces 5. 5 is a miss. Total of 3 hits.

80. What happens to the average memory access time on a machine that takes 100 cycles to access DRAM, when you move from a 32kB cache (20% miss ratio, 1 cycle cache hit or miss) to a 2MB cache (5% miss ratio, 4 cycle cache hit or miss)?
2013-Aug-Caches-6
    a. Goes up
    b. Stays the same
    c. Goes down
    d. Need more information

32kB AMAT: 80%*1+20%*(1+100)=21; 2MB AMAT: 95%*4+5%*(4+100)=9. The AMAT goes down from 21 cycles to 9 cycles.

81. Why do we have a memory hierarchy?
2013-Apr-Caches-1
    a. To make the memory look big
    b. To make the memory look fast
    c. Because we can't afford tons of SRAM
    d. All of the above

The small cache is fast for recent data, and the large memory is large for lots of data. SRAM would be great instead of DRAM, but it's too expensive and too high power.

82. What data is stored in an LRU cache?
2013-Apr-Caches-2
    a. The most important data
    b. The most recently used data
    c. The fastest data
    d. The register data

An LRU cache throws out the least recently used data, so it keeps the most recently used data.

83. What is the relative growth of memory speed vs. processor speed?
2013-Apr-Caches-3
    a. Processors are getting faster more rapidly than memory
    b. Memory is getting faster more rapidly than processors
    c. They're both getting faster at the same rate
    d. Depends on the processor

Processors are getting faster far faster than DRAM. This is because it takes so much energy to move data to/from the DRAM and very little to compute.

84. What data will be in a 4-entry, direct-mapped cache with one byte per line after the following memory accesses?
address: 0, 1, 2, 3, 4, 5, 3, 2, 1
2013-Apr-Caches-4
    a. 1, 2, 3, 4
    b. 1, 2, 3, 5
    c. 1, 3, 4, 5
    d. 1, 2, 4, 5

We fill up to 0,1,2,3, then we replace 0-->4, 1-->5, 3 hit, 2 hit, 5-->1

85. What data will be in a 4-entry, fully-associative, LRU cache with one word per line after the following memory accesses?
address: 0, 1, 2, 3, 4, 5, 3, 2, 1
2013-Apr-Caches-5
    a. 1, 2, 3, 4
    b. 1, 2, 3, 5
    c. 1, 3, 4, 5
    d. 1, 2, 4, 5

A fully-associative LRU cache will always have the most recently used data in it.

86. Which cache will have the highest percentage of data bits for a 32bit machine?
2013-Apr-Caches-6
    a. 4-entry, direct-mapped, 64 byte line
    b. 4-entry, fully-associative, 64 byte line
    c. 128-entry, direct-mapped, 64 byte line
    d. 128-entry, fully-associative, 64 byte line

The 128-entry direct-mapped cache will use 7 address bits to choose the entry in the cache, 4 bits to choose the word in the line, and 2 bits to choose the byte, so the tag will only be 19 bits.

11. Virtual Memory

1. On a computer with a modern virtual memory system, what happens when your program tries to use more data than can fit

in the physical memory?
2013-Dec-Virtual Memory-1

a. The access causes a page fault and a switch to the OS
b. The OS chooses a page to evict from DRAM and waits a painfully long time to write it to disk
c. The OS sets a PTE for your program to give you access to the now available space in memory
d. All of the above

The OS handles the fact that the page you are trying to access is not in memory by paging out another page to disk and setting up the page table to give you access to the newly freed physical page. (This might also involve loading the data for that page from disk if it is not a new page.)

2. What does it mean if we have more bits in the physical page number than we do in the virtual page number?
2013-Dec-Virtual Memory-2

a. We have very large pages
b. We have more physical memory than our ISA supports
c. We have less physical memory than our ISA supports
d. We have run out of physical memory

More bits in the physical page number than in the virtual page number means we have more physical pages (more RAM) than we have support in our ISA (virtual address space). E.g., we have >4GB of RAM on a 32-bit machine.

3. Page Table:

VPN	PPN	Permissions
0	12	read
1	disk	
2	3	COW

What happens when the program reads virtual page 2?
2013-Dec-Virtual Memory-3

a. The OS copies physical page 3 to a new page
b. The OS updates the page table to have VPN 2 point to the new page
c. The OS changes the permissions for the new page to be read/write
d. None of the above

VPN 2 is copy-on-write, but a read does not do anything other than access the data.

4. Why do we have multi-level page tables?
2013-Dec-Virtual Memory-4

a. To reduce the maximum size of the page table
b. To reduce the in-memory size of the page table
c. To make the TLB faster
d. All of the above

Multilevel page tables let us page out part of the page table as we can find it again later from the first-level page. This reduces the size of the in-memory page table.

5. What is generally slower than a page fault to a hard disk?
2013-Dec-Virtual Memory-5

a. Accessing data over the network
b. Switching to another program
c. The human using the computer
d. All of the above

Only the human is slower than accessing a hard disk.

6. What is true about virtual caches?
2013-Dec-Virtual Memory-6

a. Check the tag and TLB in parallel for speed
b. The cache can be any size regardless of associativity
c. VM protects program data in the cache from other programs
d. The TLB is not accessed on a miss

A virtual cache does not access the TLB except on a miss (to go to DRAM). As a result, there is no VM protection

for data in the cache, and the cache has to be flushed when you switch programs.

7. What would happen if two programs tried to read from the same virtual address?
   2017-Jan-Virtual Memory-1
   a. The two programs would likely crash
   b. One of them will read the correct data while the other would read random data
   c. They would read the same data
   d. They would read their own data

   Virtual memory gives the illusion to each program that it has sole access to the whole address space. If two addresses read from the same virtual address, they will read their own data, since they actually map to different physical addresses (assuming the addresses are valid in both programs).

8. What does the TLB do?
   2017-Jan-Virtual Memory-2
   a. It caches recently?used data
   b. It caches recently?used address translations
   c. It caches recently?used pages
   d. It caches recently?used instructions

   The TLB caches the recently-used virtual-to-physical address translations..

9. Why would we prefer a single-level page table rather than a multi-level one?
   2017-Jan-Virtual Memory-3
   a. To reduce the total size of the page table
   b. To make translations faster
   c. To reduce the in-memory size of the page table
   d. None of the above

   Single-level page tables are faster for lookups, so we could increase the speed of translation.

10. What does it mean if we have more bits for the virtual page number than we do for the physical page number?
    2017-Jan-Virtual Memory-4
    a. We have very large pages
    b. We have run out of physical memory
    c. We have less physical memory than our ISA supports
    d. We have more physical memory than our ISA supports

    We only have available a portion of the total number of memory that we can refer to. More bits in the virtual page number means we have more virtual memory than RAM (e.g. 64bit machine with a 4GB RAM)

11. With which TLB/cache arrangement you can have trouble with multiple applications sharing the same cache, and why?
    2017-Jan-Virtual Memory-5
    a. Virtually indexed, virtually tagged. Applications can access the same virtual address and the address cannot distinguish which one owns each cache line.
    b. Virtually indexed, virtually tagged. Applications can share memory addresses that are already cached and this can cause problems with the replacement policy.
    c. Physically indexed, virtually tagged. Need a translation to start looking in the cache and it is very slow to translate for multiple applications.
    d. Physically indexed, physically tagged. We don't need a translation to start looking in the cache and addresses for both programs can be shared, which means we don't know who is accessing the data.

    Virtually indexed and virtually tagged caches don't distinguish between two applications that access the same virtual addresses. This prevents the virtual memory system from providing protection unless some other mechanism is added such as process ID bits.

12. A program writes to VA 14 which maps to PPN (physical page number) 10 for the first time. This page is marked as COW (copy-on-write). What can you say about the next time the program accesses the same address?
    2017-Jan-Virtual Memory-6
    a. It will be in the TLB
    b. It will go to PPN 10
    c. It will not go to PPN 10
    d. It will cause a page protection fault

You can't be sure it will be in the TLB since the next access may not be for a long time. However, you can be sure it will not be to the same PPN, since COW means that the OS will have to copy the data to a new page before the program can write it.

13. Which of these problems does virtual memory not solve?
    2014-Apr-Virtual Memory-1
    a. All applications having the same 32-bit address space
    b. Not having enough physical memory for what the ISA allows
    c. Providing fast access to recently used data
    d. Preventing applications from corrupting each other

    Virtual memory does not provide fast access to recently used data. That's what a cache does.

14. What can virtual memory do for you if you have a 32-bit ISA and 8GB of physical memory?
    2014-Apr-Virtual Memory-2
    a. Nothing: a 32-bit ISA can only access 4GB of memory
    b. Not needed: a 32-bit ISA can have multiple programs access more than 4GB of memory without needing virtual memory
    c. A little: you can use the physical memory beyond 4GB instead of paging to disk for a single application
    d. A lot: you need virtual memory to have multiple 32-bit applications use more than 4GB of memory together

    A lot. Without virtual memory there is no way to access more than 4GB of physical memory. With virtual memory we can map each 32-bit application into any chunk(s) of the 8GB of physical memory and thereby use it all if we have enough applications.

15. Why do we use pages to translate virtual addresses to physical addresses?
    2014-Apr-Virtual Memory-3
    a. Pages are faster than translating individual addresses
    b. Can address more physical memory using pages
    c. Can address more physical memory with a smaller TLB using pages
    d. All of the above

    Pages allow us to have only a few TLB entries address a larger range of physical memory.

16. Which of the following pages would be good candidates to mark as copy-on-write instead of read-only?
    2014-Apr-Virtual Memory-4
    a. Compiled program code
    b. Data shared between multiple programs
    c. Library code shared between multiple programs
    d. All of the above

    You don't want to write program code (generally anything that changes compiled program code is a virus) and you don't want to change shared libraries. But you may need to be able to change shared data.

17. What size virtual memory page should you choose to address the most physical memory without a page fault on a system that supports the following TLB configuration:
    64 4kB pages
    16 2MB pages
    4 1GB pages
    2014-Apr-Virtual Memory-5
    a. 4kB pages
    b. 2MB pages
    c. 1GB pages
    d. Page size doesn't affect the amount of physical memory you can address without a page fault

    With the 1GB pages you can address 4GB without a page fault (if you're lucky and you don't access any other data). With 4KB pages you can only address 256kB and with 2MB pages only 32MB.

18. Which cache and TLB arrangement does not provide memory protection between programs?
    2014-Apr-Virtual Memory-6
    a. Virtually Indexed, Virtually Tagged
    b. Virtually Indexed, Physically Tagged
    c. Physically Indexed, Physically Tagged

d. All provide the same protection

Virtually Indexed, Virtually Tagged has no way of knowing the physical address of the data being accessed so it cannot distinguish between two programs accessing the same virtual address.

19. What does the TLB do?
`2015-Oct-Virtual Memory-1`
    a. It caches recently-used data
    b. It caches recently-used address translations
    c. It caches recently-used pages
    d. It caches recently-used instructions

The TLB caches the recently-used virtual-to-physical address translations..

20. What would happen if two programs tried to read from the same virtual address?
`2015-Oct-Virtual Memory-2`
    a. The two programs would likely crash
    b. One of them will read the correct data while the other would read random data
    c. They would read the same data
    d. They would read their own data

Virtual memory gives the illusion to each program that it has sole access to the whole address space. If two addresses read from the same virtual address, they will read their own data, since they actually map to different physical addresses (assuming the addresses are valid in both programs).

21. What are the benefits of virtual memory?
`2015-Oct-Virtual Memory-3`
    a. It increases security by isolating each process' memory
    b. It frees the applications from having to manage a shared memory
    c. It allows an application to use more memory than physically available
    d. All of the above

Virtual memory gives the illusion to each program that it has sole access to the whole address space. The advantages are multiple: It increases security by preventing an application from accessing another program's memory. It also frees applications from having to manage a shared physical memory (this burden is on the OS instead).
Finally, each applications has access to the whole virtual address space, even if there is not enough physical memory available. In this case, the OS will use secondary storage transparently.

22. What can cause a page fault?
`2015-Oct-Virtual Memory-4`
    a. Trying to read a virtual address not present in the TLB
    b. Trying to read from a page that is not in main memory but that exists on the disk
    c. Trying to read an address that has no page in the main memory and no page on the disk
    d. B and C

A fault occurs when the data is not in memory, either because it has been swapped to disk or it has not yet been read into memory or had a page allocated. If the page table entry is not in the TLB it is a TLB miss.

23. What is true if you have more physical address bits than your virtual address bits?
`2015-Oct-Virtual Memory-5`
    a. The only way to use all the physical memory is to run multiple programs
    b. You cannot address all the virtual memory from one program
    c. You can address more memory from one program than you have physical memory
    d. All of the above

The amount of physical memory that a process can address is limited by the number of virtual address bits. For example, on a system with 32-bits virtual addresses, each process can address at most 232 bytes. It means that the rest of the physical memory cannot be addressed by a single process. However, by running multiple programs, we can make use of all the physical memory available.

24. For a system with 4MB (222 bytes) pages and 32GB (235 bytes) of physical memory, figure out which bits of a 64 bits

virtual address are used for the Virtual Page Number (goes into the TLB for translation), the Page Offset and how many bits of Physical Addresses you have?

`2015-Oct-Virtual Memory-6`

a. 22 bits for PO, 42 bits for VPN. 35 bits physical addresses
b. 42 bits for PO, 35 bits for VPN. 22 bits for physical addresses
c. 35 bits for PO, 42 bits for VPN. 22 bits for physical addresses
d. 22 bits for PO, 35 bits for VPN. 42 bits physical addresses

22 bits are needed as offset within a page. The 42 bits left will be used as Virtual Page Number.
If we have 32GB of physical memory, then we need at least 35-bits long physical addresses to address all of it..

25. Why are the last 12 bits in a 4kB virtual memory page not sent through the TLB?

`2016-Aug-Virtual Memory-1`

a. Virtual memory only deals with data larger than a single page
b. They are only used in the cache tag, not the virtual memory system
c. They are used to index within the page, so they do not change even if the page is different
d. We don't send them to the TLB because the TLB is not fully-associative.

The last 12 bits indicate the address within the 4kB page. We only translate the upper bits to figure out which page, and then those last 12 bits tell us where inside that page.

26. Why would a 32-bit computer with 4GB of memory ever need to use virtual memory to store data to the disk?

`2016-Aug-Virtual Memory-2`

a. It wouldn't since 4GB is the full 32-bit address space
b. To provide security between applications
c. To allow an application to access more than 4GB of data
d. To allow multiple applications to access more than 4GB of data

If multiple applications run they can access more than 4GB total, but no one application can access more than 4GB on a 32-bit machine.

27. In a cache we use the address to look up the tag to see if the data is in the cache. What do we use to lookup in a TLB to see if the page table entry is present?

`2016-Aug-Virtual Memory-3`

a. Page offset
b. Physical page number
c. Virtual page number
d. Physical address

The virtual page number allows us to look up the physical page number. This combined with the page offset gives us the full physical address.

28. A program is processing a picture that is 3MB in memory and has been allocated in one contiguous address range. Which TLB configuration will result in the fastest performance if each time a new page table entry is brought into the TLB it takes 100 cycles? (Assume everything else is the same and that the only data we care about is in the image.)

`2016-Aug-Virtual Memory-4`

a. 32 4kB pages
b. 16 8kB pages
c. 4 1MB pages
d. All will be the same

Because the data is contiguous, we will use all the space in each page. With 4 1MB pages we can use 2 pages to access the whole picture. The other configurations will require frequent reloads of the TLB because they can't map enough address space to cover the whole image.

29. What benefit would virtual memory give you on a 32-bit MIPS machine with 4GB of memory that only runs one program at a time?

`2016-Aug-Virtual Memory-5`

a. Protection to prevent data corruption
b. Access to the full 32-bit address space
c. Flexibility for using the physical memory
d. None

With only one program and 4GB of memory, virtual memory provides no benefits. That one program already has full access to the full 32-bit address space and no other programs are running to corrupt its data.

30. Why is a Virtually Indexed, Physically Tagged cache a good design?
    2016-Aug-Virtual Memory-6
    a. You can access the TLB and the cache tags in parallel to save time.
    b. The cache index address is unique across programs to provide security.
    c. Using separate types of addresses allows the TLB and cache to be smaller so they are faster.
    d. All of the above.

    VIPT allows us to do the translation (VA->PA) a the same time as we look up the tag (VA->PA tag) and then after both happen in parallel we can compare them. This avoids having to do the VA->PA TLB translation first before we can do a PA->PA tag lookup..

31. Which of the following statements is true about virtual memory?
    2014-Dec-Virtual Memory-1
    a. VM provides security by preventing programs from changing each other's data, flexibility by letting programs use as much memory as is physically available, and performance by making DRAM accesses faster
    b. VM provides security by preventing programs from changing each other's data, flexibility by letting programs use as much memory as is physically available, and performance by allowing programs to share data without copying it.
    c. VM provides security by preventing programs from changing each other's data, flexibility by letting programs use more memory than is physically available, and performance by allowing programs to share data without copying it.
    d. All statements above are true.

    Virtual memory does not make DRAM accesses faster. Indeed it makes them slower because we need to do the virtual to physical translation. VM allows to access more memory than the available physically.

32. What is the average CPI (cycles per instruction) of a program that is 45% load/store instructions, and runs on a 1 CPI processor with a perfect cache, and a perfect TLB that takes 3 cycles per translation?
    2014-Dec-Virtual Memory-2
    a. 0.35
    b. 1.35
    c. 3.35
    d. 5.35

    Every instruction needs 1.45 translations (1 for the instruction and 0.45 for the data). Normally an instruction takes 1 cycle, but now it takes 1*3 for the TLB +0.45*3 for the data TLB + 1 for the execution = 5.35

33. A program is executed using a processor with a TLB that has 32 entries and the cache is 64KB with a 64 byte line size. How much data can the program access without a TLB miss?
    2014-Dec-Virtual Memory-3
    a. 64kB
    b. 512kB
    c. 2MB
    d. Don't have enough information

    You need to know the size of each page to calculate the TLB reach. The size of the cache is irrelevant.

34. What part of this information from PowerPoint's page table tells you whether this data is shared?

```
START - END             [VSIZE  RESDNT] PRIV    SHRMOD  REGION DETAIL
11d67000-11d89000       [  136K    44K] rw-     SM=COW  Fonts/Comic SansMS.ttf
```

    2014-Dec-Virtual Memory-4
    a. Virtual size is less than resident size so it is was loaded by another program and must be shared
    b. The privileges are not read-only so it cannot be shared
    c. The sharing mode is COW so it will be copied if needed, which is only necessary for shared data
    d. The memory region is in the shared address range

    The COW indicates that it is shared since we could otherwise just write to the copy we have without worrying about other programs.

35. How could a system with 32-bits of virtual address use 33 bits of physical address space?

2014-Dec-Virtual Memory-5

a. It couldn't
b. Each application could use more than 32-bits of address space
c. Multiple applications together could use more than 32-bits of address space
d. No application could use the full virtual address space

Having 33 physical address bits for a 32-bit system means that each application can only address 4GB (2^32) of memory (its full virtual address space) but the system has 8GB (2^32) of installed memory. The system can use this memory by giving multiple applications more memory, but cannot give any one 32-bit application more than 4GB, since it could not address it.

36. What happens when an application with the following page table tries to execute this code:

```
int data[2048]; // starts at address 4096. An int is 1 word.
for (int i=0; i<2048; i++)
        data[i]=data[i]+1;
Page table. (4kB pages)
VPN 0 -> PPN 12 read only
VPN 1 -> PPN 12 read&write
VPN 2 -> PPN 64 read only
VPN 3 -> PPN 34 CoW
```

2014-Dec-Virtual Memory-6

a. Runs as expected
b. The OS copies part of the data when it is written
c. The program crashes due to a page protection violation
d. Can't tell

The array data starts at address 4096, which is the beginning of VPN 1, and is 2048 words long, or 8192 bytes, which is two 4096 byte pages. So the first half is in VPN 1 which is read and write, but the second half is in VPN 2, which is read only. When the program tries to write to that second half it will cause a page protection violation and crash.

37. How much data can a program access without a TLB miss on a machine with a 64 entry, 8-way set associative TLB where each page is 4kB, and a 16kB cache with a 64 byte line size.

2015-Apr-Virtual Memory-1

a. 16kB
b. 256kB
c. 1MB
d. Don't have enough information

With 64 TLB entries in total and a 4kB page you could access 64*4kB=256kB of data without needing to update the TLB. (Not all of them would be hits if the cache was only 16kB, though.)

38. What happens before the actual write to DRAM when a program with the following page table tries to write to address 14?

```
VA-->PA         on disk         access bits
2-->8           0               read/copy on write
1-->9           0               read/write
14-->7          0               read/write
16-->14         1               read/write
13-->6          1               read only
```

2015-Apr-Virtual Memory-2

a. VA-->PA translation, then the data is loaded from the disk
b. VA-->PA translation, then a copy on write
c. VA-->PA translation, then a memory protection exception is generated
d. VA-->PA translation

The TLB has an entry for virtual address 14, so the TLB translates VA 14 to PA 7 before writing to physical address 14.

39. Writing to which address with the following page table would result in a page being loaded from disk?

```
VA-->PA         on disk         access bits
2-->8           0               read/copy on write
1-->9           0               read/write
14-->7          0               read/write
16-->14         1               read/write
13-->6          1               read only
```

2015-Apr-Virtual Memory-3

a. 8
b. 13
c. 14
d. 16

Writing to address 8 would result in memory protection exception/segmentation fault since that address is not in the page table. Address 13 would result in a segmentation fault (since you don't have permission), address 14 would result in a direct write to DRAM since the page is in memory, but address 16 requires that the data be loaded from disk before it can be written.

40. What can virtual memory do for you if you have an ISA with a 64-bit address space and 16GB of physical memory?
2015-Apr-Virtual Memory-4
    a. Nothing: a 64-bit address space can address a lot of memory already so it is not useful
    b. Not needed: a 64-bit address space can have multiple programs access more than 16GB of memory without needing virtual memory
    c. A little: you can use the physical memory beyond 16GB instead of paging to disk for a single application
    d. A lot: you need virtual memory to keep multiple applications from crashing each other.

Virtual memory allows us to have multiple applications each with their own private memory space with protection guarantees from the OS.

41. Why is a Virtually Tagged, Physically Indexed cache bad?
2015-Apr-Virtual Memory-5
    a. Slow: have to do the translation before you can access the cache.
    b. Insecure: no way to prevent different programs from sharing data in the cache.
    c. Both A and B
    d. It's not: it is both fast and secure

A VTPI cache requires a translation before you can index into and then you get back a virtual tag, which doesn't help you keep applications' virtual address spaces separate.

42. The TLB operates like a cache for page table entries. Which of the following does a TLB have that a data cache does not?
2015-Apr-Virtual Memory-6
    a. A replacement policy
    b. Dirty bits
    c. Permission bits
    d. Tag bits

The TLB needs to have permission bits to keep track of what the program is allowed to do with each page. Regular caches allow the program to do anything it wants with the data it finds, so it does not need permission bits. Both need dirty bits to know what to do with the data (cache) or page (TLB) if it is evicted.

43. What is not a benefit of having a virtual address to physical address translation?
2013-Jan-Virtual Memory-1
    a. Memory protection
    b. Using disk as memory
    c. Faster cache accesses
    d. Using fragmented memory

44. If each page table entry maps one word and requires 4 bytes, how large is the whole page table for a 32-bit machine?
2013-Jan-Virtual Memory-2
    a. 4 bytes
    b. 4kB
    c. 4MB
    d. 4GB

45. If a machine does not translate 13 bits of the virtual address, how large are its pages?
    2013-Jan-Virtual Memory-3
    a. 4kB
    b. 8kB
    c. 2MB
    d. 2GB

46. Do page tables need a dirty bit?
    2013-Jan-Virtual Memory-4
    a. No, they don't store data, just VA-to-PA mappings
    b. No, they are read only
    c. Yes, the bit indicates if the data is on disk
    d. Yes, the bit indicates if the page has been changed since it was loaded from disk

47. What is the problem with a virtually indexed, virtually tagged cache?
    2013-Jan-Virtual Memory-5
    a. Have to access the TLB before the cache lookup
    b. Cache size is limited by the page size
    c. Need extra bits to tell which processes owns each line
    d. Processes can not share data

48. What happens when process 1 loads address 10 given the page tables below?

process	VA	PA	valid	dirty
1	3	4	1	1
1	6	5	1	0
0	3	6	1	0
0	10	7	1	1

    2013-Jan-Virtual Memory-6
    a. It loads physical address 7 from DRAM
    b. It loads physical address 10 from DRAM
    c. It causes a page fault and the OS loads the data for physical address 7 from disk before returning it to the program
    d. It causes an protection violation and the OS kills it

49. How does virtual memory help with security?
    2015-Aug-Virtual Memory-1
    a. Provide each program a full 32-bit address space
    b. Prevent programs from accessing other programs' physical memory
    c. Allow the OS to place a program's data anywhere in physical memory
    d. All of the above

    Virtual memory makes sure that when two programs try to load the same (virtual) address they get different physical data, so they cannot access each other's data or corrupt each other.

50. What would happen if two programs ran together without virtual memory and accessed the following data?
    Program A:
    read-only: 0x000-0x020
    read/write: 0x100-0x200, 0x350-0x400
    Program B:
    read-only: 0x000-0x080
    read/write: 0x080-0x90, 0x220-0x320, 0x800-0x950.
    2015-Aug-Virtual Memory-2
    a. They would crash
    b. They would run fine
    c. They would run out of memory
    d. They could not access their full 32-bit address space

    These two programs don't overwrite each other anywhere, so they would run fine. The programs can each access the full 32-bit address space, but doing so might cause them to crash if they corrupt each other's data.

51. What does it mean if we have fewer bits in the virtual page number than we do in the physical page number?
    2015-Aug-Virtual Memory-3
    a. We have small pages

b. We have run out of physical memory
c. We have less virtual memory space than physical memory space
d. We have more virtual memory space than physical memory space

More bits in the physical page number means we have more physical pages than virtual pages. This means we have more physical memory than our ISA supports for its virtual addresses. This can happen if you are running a 32-bit program on a machine with more than 4GB of physical memory.

52. A program writes to VA 14 which maps to PPN (physical page number) 10 for the first time. This page is marked as COW (copy-on-write). What can you say about the next time the program accesses the same address?
`2015-Aug-Virtual Memory-4`
    a. It will be in the TLB
    b. It will go to PPN 10
    c. It will not go to PPN 10
    d. It will cause a page protection fault

You can't be sure it will be in the TLB since the next access may not be for a long time. However, you can be sure it will not be to the same PPN, since COW means that the OS will have to copy the data to a new page before the program can write it.

53. What is/are the biggest reasons that are page faults so excruciatingly slow?
`2015-Aug-Virtual Memory-5`
    a. The TLB has to be reloaded with the right page table data
    b. The data has to be accessed from the hard disk
    c. The OS has to choose which data to evict from DRAM
    d. All of the above

While all of the above are true, the vast majority of the performance loss comes from accessing the data on the hard disk. The others are very fast in comparison.

54. If you use a much faster flash drive (NVRAM) instead of a spinning hard disk, what changes about virtual memory? Accessing the data on the drive is now so much faster that:
`2015-Aug-Virtual Memory-6`
    a. We need to optimize the TLB to be faster on hits
    b. Page faults are no longer a performance problem
    c. We need to optimize how quickly the OS can choose which page to evict.
    d. We no longer need to evict data from DRAM if we run out of physical pages.

Previously we could spend as long as we want in the OS choosing a page to evict because it took so long to access the hard disk. If we now have a much faster disk (Flash) then we need to be careful to not spend too much time in the OS.

55. What is *not* a benefit of virtual memory?
`2014-Sep-Virtual Memory-1`
    a. Memory is protected from other applications
    b. Memory accesses are faster
    c. Memory appears larger than the installed DRAM
    d. Memory does not have to be used in contiguous chunks

Memory accesses with virtual memory first have to be translated, which can be very slow if the address is not in the TLB.

56. What size virtual memory pages should you choose to address the most physical memory without a page fault on a system that supports the following TLB configuration: 128 pages with 8kB pages, or 8 pages with 256MB pages, or 1 page with 1GB pages.
`2014-Sep-Virtual Memory-2`
    a. 8kB pages
    b. 256MB pages
    c. 1GB pages
    d. Page size doesn't affect the amount of physical memory you can address without a page fault

The amount of memory that can be accessed by the TLB without a translation fault is the (page size)*(number of pages). With 8*256MB pages you can address 2GB.

57. With which TLB/cache arrangement you can have trouble with multiple applications sharing the same cache, and why?
2014-Sep-Virtual Memory-3
    a. Virtually indexed, virtually tagged. Applications can access the same virtual address and the address cannot distinguish which one owns each cache line.
    b. Virtually indexed, virtually tagged. Applications can share memory addresses that are already cached and this can cause problems with the replacement policy.
    c. Physically indexed, virtually tagged. Need a translation to start looking in the cache and it is very slow to translate for multiple applications.
    d. Physically indexed, physically tagged. We don't need a translation to start looking in the cache and addresses for both programs can be shared, which means we don't know who is accessing the data.

    Virtually indexed and virtually tagged caches don't distinguish between two applications that access the same virtual addresses. This prevents the virtual memory system from providing protection unless some other mechanism is added such as process ID bits.

58. If two programs want to share data using virtual memory, they can do so by mapping some pages to the same physical memory. On which type of architectures does this not work?
2014-Sep-Virtual Memory-4
    a. On machines with a virtually indexed, physically tagged cache
    b. On machines with a virtually indexed, virtually tagged cache
    c. On machines with a physically indexed, virtually tagged cache
    d. It will work on all of the above

    This approach will not work on a virtually indexed, virtually tagged cache because the shared data is mapped to the same physical address, but not necessarily the same virtual address in each application. This means that if the cache is accessed purely by the virtual address that each application can have its own distinct virtually-addressed copy and they won't share as expected.

59. What does it mean if we have more bits for the virtual page number than we do for the physical page number?
2014-Sep-Virtual Memory-5
    a. We have very large pages
    b. We have run out of physical memory
    c. We have less physical memory than our ISA supports
    d. We have more physical memory than our ISA supports

    We only have available a portion of the total number of memory that we can refer to. More bits in the virtual page number means we have more virtual memory than RAM (e.g. 64bit machine with a 4GB RAM)

60. Why would we prefer a single-level page table rather than a multi-level one?
2014-Sep-Virtual Memory-6
    a. To reduce the total size of the page table
    b. To make translations faster
    c. To reduce the in-memory size of the page table
    d. None of the above

    Single-level page tables are faster for lookups, so we could increase the speed of translation.

61. For a system with 2MB (2^{21} bytes) pages and 16GB (2^{34} bytes) of physical memory, figure out which bits of a 64 bits virtual address are used for the Virtual Page Number (goes into the TLB for translation), the Page Offset and how many bits of Physical Addresses you have?
2016-Oct-Virtual Memory-1
    a. 43 bits for PO, 21 bits for VPN, 34 bits physical addresses
    b. 34 bits for PO, 43 bits for VPN, 21 bits physical addresses
    c. 21 bits for PO, 43 bits for VPN, 34 bits physical addresses
    d. 34 bits for PO, 21 bits for VPN, 43 bits physical addresses

    21 bits are needed as offset within a page. The 43 bits left will be used as Virtual Page Number. If we have 16GB of physical memory, then we need at least 34 bits long physical addresses to address it all.

62. Why would you use larger pages?
2016-Oct-Virtual Memory-2
    a. It reduces the number of cache misses
    b. It allows your application to use more memory

c. It reduces how long it takes to handle each page fault (load a TLB entry)
d. It increases the TLB reach

Increasing the page size means that each TLB entry covers a larger range of memory, thus potentially reducing the number of TLB misses and increasing

63. Why would you not use larger pages?
`2016-Oct-Virtual Memory-3`
    a. Each TLB access takes more time
    b. It can waste memory for programs that use small chunks of data
    c. It makes page faults to disk take longer
    d. B and C

Large pages can waste memory if the pages allocated are sparsely populated. Moreover, major page faults (page faults where we need to swap data to disk) will take more time, since the amount of data to copy to/from the disk is bigger

64. What would happen if two applications tried to read the virtual address 0xffff0 in parallel?
`2016-Oct-Virtual Memory-4`
    a. They would read the same data
    b. They would read their own data
    c. It depends on the timing: The first application to execute will successfully read data, while the second will crash
    d. It depends on where each application's page tables maps the virtual address

Virtual memory gives the illusion to each program that it has sole access to the whole address space. If two applications read from the same virtual address, they will read their own data unless those addresses are mapped to the same physical address. So you need to know how the page table maps them to answer.

65. A program is executed on a processor with a TLB that has 64 entries and a page size of 2MB. The processor also has a 2MB L3 cache with a cache line size of 128 bytes. How much data can the program access without a TLB miss?
`2016-Oct-Virtual Memory-5`
    a. 128 bytes
    b. 2MB
    c. 64MB
    d. 128MB

The TLB has 64 entries, each pointing to a 2MB page. The TLB-reach is thus 64MB*2 =128MB. The cache configuration is irrelevant.

66. For a system with 32-bits virtual addresses and 64GB of RAM, which of the following statements are true?
`2016-Oct-Virtual Memory-6`
    a. 60GB of RAM will be unusable
    b. Each application can only use at most 4GB of memory
    c. It is possible for the system to use the whole memory despite the small virtual address space
    d. B and C

Each application has a 32-bits address space, which means that each application can only use at most 4GB of memory. However, if we run multiple applications, each of them will be able to use 4GB of memory, and it thus possible to use all the physical memory available.

67. How does virtual memory prevent two copies of the same program, run at the same time, from overwriting each other?
`2016-Jan-Virtual Memory-1`
    a. It changes the offset for the programs so their instructions use different addresses
    b. It changes where the addresses go in memory so the programs use different data
    c. It only allows different programs to run together so their addresses don't overlap
    d. It doesn't: two copies of the same program will access the same memory addresses

Virtual memory provides a level of indirection whereby the program's addresses (VA) go to different locations in memory (PA) so each program accesses different data, even if it uses the same addresses for the instructions..

68. What is the benefit of virtual memory for a 32-bit computer with more than 4GB of memory?
`2016-Jan-Virtual Memory-2`

a. Individual programs can access more than 4GB of memory
b. Programs are protected from each other
c. Prevents holes in used memory when programs quit
d. All of the above

A 32-bit computer can only access 4GB of memory per program (that's the VA size). Virtual Memory allows you to have holes in memory and fill them in with other programs. VM provides protection, even if it can't provide more VA space..

69. Computers typically use 4kB pages, but often support larger (2MB and 1GB) pages as well. What is the advantage of larger pages for programs that access data sequentially?
2016-Jan-Virtual Memory-3
    a. Fewer TLB misses
    b. Fewer translations
    c. Can access more memory
    d. All of the above

Larger pages allow the program to access more data for each TLB entry, which reduces the number of TLB misses..

70. Which cache configuration requires both serial access to the TLB and cache and does not provide protection?
2016-Jan-Virtual Memory-4
    a. Virtually Indexed, Physically Tagged
    b. Physically Indexed, Physically Tagged
    c. Physically Indexed, Virtually Tagged
    d. Virtually Indexed, Virtually Tagged

Privately Indexed requires that you access the TLB before the cache (you need the physical tag). Virtually Tagged does not provide protection since the data is stored by the virtual address. Therefore PIVT is the worst combination of both..

71. What is the TLB?
2016-Jan-Virtual Memory-5
    a. A cache
    b. A predictor
    c. A tag
    d. A page table

The TLB caches the most recently used page table entries to speed up translation.

72. The page table for two programs is shown below. Which address should program 1 use in its sw instruction to share data with program 2?

```
Program 1 (VA-->PA)          Program 2
1 --> 5                      1 --> 7
2 --> 6                      3 --> 4
3 --> 7                      7 --> 3
7 --> 1                      6 --> 2
```

2016-Jan-Virtual Memory-6
    a. 1
    b. 3
    c. 2
    d. 7

The only shared PA is 7. Program 1 writes to VA 3 and the data goes to PA 7. Program 2 can then access it via VA 1, which will access the same PA 7.

73. Virtual memory does not do which of the following?
2013-Aug-Virtual Memory-1
    a. Provide security by preventing programs from changing each other's data
    b. Provide flexibility by letting programs use more memory than is physically available
    c. Provide flexibility by letting the OS put programs anywhere in physical memory
    d. Provide performance by making DRAM accesses faster

Virtual memory doesn't make DRAM accesses faster. Indeed it makes them slower because first we need to do the virtual to physical translation.

74. How much data can a program access without a TLB miss if the TLB has 8 entries and the cache is 16kB with a 64 byte line size.
`2013-Aug-Virtual Memory-2`
    a. 512 bytes
    b. 16kB
    c. 32kB
    d. Don't have enough information

You need to know the size of each page to calculate this. The size of the cache is irrelevant. With 8 TLB entries and a 4kB page you could access 8*4kB=32kB of data without needing to update the TLB. (Not all of them would be hits if the cache was only 16kB.) But if you had 2MB pages then you could access 8*2MB=16MB of data.

75. What is the average CPI of a program that is 33% load/store instructions, run on a 1 CPI processor with a perfect cache, and a perfect TLB that takes 2 cycles per translation?
`2013-Aug-Virtual Memory-3`
    a. 0.66
    b. 1.66
    c. 2.66
    d. 3.66

Every instruction needs 1.33 translations (1 for the instruction and 0.33 for the data). Normally an instruction takes 1 cycle, but now it takes 1*2 for the instruction TLB +0.33*2 for the data TLB +1 for the execution = 3.66.

76. Why is a virtually indexed, physically tagged cache a good design?
`2013-Aug-Virtual Memory-4`
    a. You can do the translation and cache lookup in parallel
    b. The size of the cache is limited by the associativity and page size
    c. You don't need the translation to determine if you have a cache hit
    d. All of the above

You can use the virtual address to look into the cache and extract the tag while you translate the address at the same time. Then when the cache is done with the lookup the translation is done, so you were able to do them in parallel.

77. Why is a virtually indexed, virtually tagged cache a good design?
`2013-Aug-Virtual Memory-5`
    a. You can do the translation and cache lookup in parallel
    b. The size of the cache is limited by the associativity and page size
    c. You don't need the translation to determine if you have a cache hit
    d. All of the above

Virtually-index/virtually-tagged caches let you access the cache directly with the virtual address so you don't need to do a translation unless you have a miss.

78. Two programs want to share data. Can this be done with virtual memory?
`2013-Aug-Virtual Memory-6`
    a. No, each program has its own virtual address space that maps to its own physical memory.
    b. Yes, they can both have pages that map to the same physical memory.
    c. Only if they use a physically-index and physically-tagged cache.
    d. Only if they use a virtually-index and virtually-tagged cache.

Programs can share data by having the OS map their virtual addresses to the same physical page. This does cause problems for the cache, but with a combination of flushing and process ID tags these can be overcome.

79. How does virtual memory make software more portable?
`2013-Apr-Virtual Memory-1`
    a. It allows programs to be put at different places in memory
    b. It allows programs to use more memory than is available
    c. It allows programs to separated from each other
    d. All of the above

The memory translation allows us to put programs at different physical addresses, use more memory than we have installed, and prevent programs from accessing each other's memory.

80. What happens when the program with the following page table tries to write to address 14?

```
VA --> PA       on disk         access bits
1 --> 8         0               read/write
2 --> 9         0               read/write
3 --> 14        0               read only
13 --> 15       1               read/write
14 --> 7        1               read only
```

2013-Apr-Virtual Memory-2

a. The data is loaded from the disk
b. The PTE is loaded from the page table into the TLB
c. An memory protection exception is generated
d. The data is written to physical address 7

The data is marked as read only and on disk. So trying to write it will generate a memory protection exception. (It won't be read in from disk unless the exception is first handled.)

81. Which TLB/cache arrangement allows you to know if you have a hit in the cache without checking the TLB?

2013-Apr-Virtual Memory-3

a. Virtually indexed, virtually tagged
b. Virtually indexed, physically tagged
c. Physically indexed, virtually tagged
d. Physically indexed, physically tagged

Virtually indexed means you don't need the TLB translation to start looking in the cache, and the virtual tag means you can see if the hit is what you are looking for without needing the physical translation.

82. Which TLB/cache arrangement allows you to access the TLB in parallel with the cache? (And requires it on every access.)

2013-Apr-Virtual Memory-4

a. Virtually indexed, virtually tagged
b. Virtually indexed, physically tagged
c. Physically indexed, virtually tagged
d. Physically indexed, physically tagged

Virtually indexed means you don't need the TLB translation to start looking in the cache, but physically tagged means you do need it to see if the data in the cache is what you are looking for.

83. What do you do when you can't find a PTE in the TLB?

2013-Apr-Virtual Memory-5

a. Load the page from the cache
b. Load the page from disk
c. Look in the full page table in memory
d. Cause a memory protection exception

If the page isn't in the TLB, then we need to look in the full page table in memory.

84. Which data is shared between processes 1 and 2?

```
Process 1 PT (VA--> PA)         Process 2 PT  (VA--> PA)
1 --> 12                        14 --> 4
2 --> 13                        13 --> 6
3 --> 24                        3  --> 3
7 --> 5                         4  --> 25
8 --> 3                         5  --> 7
```

2013-Apr-Virtual Memory-6

a. P1 8 shares with P2 3
b. P1 3 shares with P2 3
c. P1 7 shares with P2 5
d. No sharing

P1 VA 8 is mapped to the same PA (3) as P2 VA 3. Therefore they share the same memory.

12. Parallelism

1. Why do we do parallelism?
   `2013-Dec-Parallelism-1`
   a. To reduce power
   b. To overcome Moore's law
   c. For performance
   d. Because it's fun

   Parallelism can be more power-efficient, but it doesn't in itself reduce power. We certainly don't do it because it is fun. We only do it for performance.

2. What does having a lock for a variable do?
   `2013-Dec-Parallelism-2`
   a. Prevents other processors from changing it
   b. Lets other processor (who bother to check) know that they shouldn't change it
   c. Updates the variable in other processors' caches
   d. Updates the variable in my processor's cache

   The lock is just an indication to other processors to not touch the data. They have to actually check (and respect) the lock for this to be useful.

3. If my program is 80% parallel and I have 100,000 processors, how much faster can I make the program?
   `2013-Dec-Parallelism-3`
   a. 100,000x
   b. 80,000x
   c. 5x
   d. 1.8x

   I can make the 80% take virtually zero time, but I've still got the 20% that isn't parallel left. So I can run it in 20% of the time, or only 5x faster.

4. Is it hard to get locking in a program right?
   `2013-Dec-Parallelism-4`
   a. No
   b. Somewhat
   c. Very (people have died because of locking errors)
   d. No one uses locks anymore

   Getting locks right is often incredibly tricky, largely because it is hard to debug. (The errors only happen when the processors access the data in just the right order.)

5. Why do we need cache snooping and cache coherency?
   `2013-Dec-Parallelism-5`
   a. To make sure that the local processor's cache data isn't changed by other processors
   b. To update the dirty bits across multiple caches
   c. To make sure every cache has the latest value written to each address in any cache
   d. All of the above

   Snooping and coherency ensure that each processors sees the latest writes to every piece of data so it doesn't accidentally use the local version instead of the newer version written by another processor.

6. The Intel Nehalem processor has 4 processors (cores) and 32kB of L1 cache for each processor, 256kB of L2 cache for each processor, and 8MB of L3 cache shared between all 4 processors. The Intel Westmere processor has 12MB of L3 cache. What happens when you run 4 programs that each use 3MB of data on the Nehalem vs. the Westmere? (Assume the only difference is the cache size.)
   `2013-Dec-Parallelism-6`
   a. No difference
   b. The programs will run slower on the Nehalem than on the Westmere
   c. The programs will run faster on the Nehalem than on the Westmere

d. Some of the programs will crash on the Nehalem but all will run on the Westmere

The programs can all fit their data into the cache on the Westmere but not on the Nehalem. Since the cache is so much faster than the main memory, the programs will run much faster on the Westmere.

7. What do you *need* to add to the pipeline to support executing load/store instructions in parallel with other instructions?
`2017-Jan-Parallelism-1`
   a. Extra data memory
   b. Extra data memory ports
   c. Extra register file ports
   d. All of the above

We don't need to add extra data memory or data memory ports. We only need to add ports to read from the register files to read from/write to.

8. Which of the following statement is *incorrect* about locks and a particular piece of data?
`2017-Jan-Parallelism-2`
   a. Locks don't prevent any other processor from changing the data while it is locked.
   b. Locks don't prevent any other processor from accessing the data while it is locked.
   c. Locks enable each processor to safely update the data at the same time.
   d. Locks enable the program to keep track of when other processors are access the data.

Locks only serve to indicate what is going on; they don't allow multiple processors to use the data at the same time without problems.

9. A program takes 200s to run on a single CPU, and 35% of it can be run in parallel. How long will it take to run the same program on a multicore CPU with 8 cores?
`2017-Jan-Parallelism-3`
   a. 112s
   b. 120s
   c. 138s
   d. 145s

Amdahl's law gives us a maximum speedup of 1 / (0.65 + 1 / 8 * 0.35) = 1.44x, which means the program will complete in 138s instead of 200s..
system over the whole execution is 66GFlops.

10. How much faster can my program run if I have 9000 cores and 10% of my program cannot be parallelized?
`2017-Jan-Parallelism-4`
   a. 9000x faster
   b. 4500x faster
   c. 5x faster
   d. 10x faster

Using Amdahl's law, we get a speedup of 1 / (0.10 + 1/9000 * 0.90) = 10x

Another way to answer this question is to say that by having infinite parallelism, 90% of this program will be done in 0 unit of time. Only 10% of it must be done serially, and thus the total execution time is divided by 10..

11. Why do we do parallelism?
`2017-Jan-Parallelism-5`
   a. Faster
   b. Easier
   c. Lower power
   d. All of the above

Parallel code and hardware is (a lot) harder than serial code and hardware. Parallel execution can be more power-efficient than serial, but it doesn't in itself reduce power. We only put up with the pain and complexity of parallelism for performance.

12. What happened in 2003/2004 that led us to have multicore chips everywhere today?
`2017-Jan-Parallelism-6`

a. Processors got small enough that you could put multiple cores on the same chip to get better performance
b. Processors got so hot that it was too expensive to cool them, so manufacturers switched to putting multiple slower (cooler) processors on a chip
c. Manufacturers couldn't increase the clock speed, so they put multiple cores on the same chip to get better throughput
d. All of the above

Processors never really got smaller: people kept adding more and more features so they were getting larger even faster than the rate at which transistors were getting smaller. Manufacturers could indeed increase the clock speed (Intel did). The killer was power: it was too expensive to cool faster chips so they switched to multicore and we've been suffering ever since.

13. What does a lock do?
`2014-Apr-Parallelism-1`
    a. Protect data so only one processor can access it at a time
    b. Synchronize data so that different caches see the same value
    c. Provide a way for a program to check if another processor is using the data
    d. All of the above

All locks do is provide a way for a program to check if another processor doing something. They don't actually fix any synchronization, but they allow you to write a program that is synchronized.

14. Why did manufacturers start putting multiple cores on each chip in 2004?
`2014-Apr-Parallelism-2`
    a. It gave users better performance
    b. They couldn't make the clocks any faster
    c. It was too expensive to cool chips at faster speeds
    d. All of the above

Most users did not get better performance because they did not have parallel programs. They could absolutely make the clocks faster, but they couldn't afford to cool them if they did.

15. If I have 10 processor cores and 90% of my program is perfectly parallel, in what percentage of the original time will my program run?
`2014-Apr-Parallelism-3`
    a. 1.9%
    b. 10%
    c. 19%
    d. 90%

The 90% will go 10x faster, so that will go down to 9% and you have the remaining 10% that is not parallel for 19%.

16. Why do we need atomic instructions for building a lock?
`2014-Apr-Parallelism-4`
    a. Need to make sure that no one interrupts us from the time we see the lock is available to the time we have acquired it
    b. Need to avoid hazards in the pipeline to avoid synchronization issues with other instructions in our program
    c. Need to make sure data in the cache is shared with all other processors
    d. All of the above

Atomic instructions make sure we can both check if we can take the lock and take it without being interrupted. Pipeline synchronization issues are handled by the forwarding logic and making sure the data is shared is handled by the coherency logic.

17. Improper uses of locks has resulted in what?
`2014-Apr-Parallelism-5`
    a. Program crashes
    b. Incorrect results
    c. Deaths
    d. All of the above

Programs can easily crash and produce incorrect results if multiple processors change the data at the same time.

This has resulted in many deaths when these programs interact with the real world, including massive power outages and medical overdoses.

18. Does having write-through caches for each processor mean you do not need to check for coherency between them?
`2014-Apr-Parallelism-6`
    a. Yes, when data is changed it always goes back to the main memory
    b. Yes, when data is changed it always goes to all the caches
    c. No, when data is changed there may still be old values in other caches that are just reading it
    d. No, when data is changed it only updates the local cache

If you have two processors with write-through caches and they each have copies of the data in their cache, then one processor writing to the cached data will update the main memory (write-through) but that will not update the data in the other cache.

19. A program takes 200s to run on a single CPU, and 35% of it can be run in parallel. How long will it take to run the same program on a multicore CPU with 8 cores?
`2015-Oct-Parallelism-1`
    a. 112s
    b. 120s
    c. 138s
    d. 145s

Amdahl's law gives us a maximum speedup of 1 / (0.65 + 1 / 8 * 0.35) = 1.44x, which means the program will complete in 138s instead of 200s..

20. Which of the following designs to improve performance does not require multiple PCs?
`2015-Oct-Parallelism-2`
    a. Simultaneous multithreading (issuing instructions from multiple programs into the pipeline at the same time)
    b. Multicore processors (having multiple processors on one chip)
    c. Dual-issue processors (issuing multiple instructions from the same program into the pipeline at the same time)
    d. All of them require multiple PCs

Dual-issue processors issue two instructions each cycle, so the PC simply counts up by 8. The others require multiple PCs to keep track of which instruction in each program is next.

21. Why do we need locks?
`2015-Oct-Parallelism-3`
    a. To synchronize accesses to the same data by different processors
    b. To make sure only one thread can use the pipeline at the same time
    c. To avoid page faults
    d. None of the above

Locks ensure that data accessed by different processors or programs are properly synchronized..

22. The Intel Haswell Processor (CPU) with the Iris Pro 5100 integrated Graphics Processor (GPU) has the following characteristics: 40 GPU cores that can each execute 20.8B floating point operations per second per core and 4 CPU cores that can each execute 56B floating point operations per second per core. Which is better for a program that is 20% parallel?
`2015-Oct-Parallelism-4`
    a. Run on the CPU cores
    b. Run on the GPU cores
    c. Same
    d. Need more information

On the GPU, Amdahl's law gives us a speedup of 1.24x, thus the average speed of the system over the whole execution is 26GFlops

On the CPU, Amdahl's law gives us a speedup of 1.18x, thus the average speed of the system over the whole execution is 66GFlops.

23. How much faster can my program run if I have 9000 cores and 10% of my program cannot be parallelized?
`2015-Oct-Parallelism-5`
    a. 9000x faster
    b. 4500x faster

c. 5x faster
d. 10x faster

Using Amdahl's law, we get a speedup of 1 / (0.10 + 1/9000 * 0.90) = 10x

Another way to answer this question is to say that by having infinite parallelism, 90% of this program will be done in 0 unit of time. Only 10% of it must be done serially, and thus the total execution time is divided by 10..

24. An image processing algorithm divides an image in 4 equally-sized parts and works on each part on a separate CPU in parallel. Each processor completes its work in 10 seconds, except one that completes in 12 seconds. If we run this program on a CPU with 8 cores, what is its execution time?
`2015-Oct-Parallelism-6`
    a. 2 seconds
    b. 10 seconds
    c. 12 seconds
    d. 5.25 seconds

As long as we have at least as many cores as the program has threads, the execution time of this program will be the execution time of the slowest thread.
Moreover, having more cores than threads is of no help here (4 cores would just idle).

25. Which of the following processor designs can execute more than one instruction at the same time?
`2018-Oct-Parallelism-1`
    a. Pipelined Single-issue
    b. Pipelined Dual-issue
    c. Pipelined Symmetric Multithreading (SMT)
    d. All of the above

A pipelined processor processes multiple instructions at the same time across the stages (pipeline parallelism). Dual-issue means it can do twice as many at the same time. SMT, however, just means it can switch between instructions from different programs, but it may or may not do them at the same time.

26. If a program is 90% parallel and we execute it using 1M processors, how much faster will it run?
`2018-Oct-Parallelism-2`
    a. 1,000,000x faster
    b. 900,000x faster
    c. 10x faster
    d. 1.9x faster

We will make 90% of the program 1M times faster (essentially zero time) but the remaining 10% will be just as slow. This means we get a 10x speedup.

27. What does a lock do?
`2016-Aug-Parallelism-1`
    a. Provide a way for a program to check if another processor is using the same data
    b. Provide a way for a program to prevent another program from using the same data
    c. Synchronize accesses between two programs so they don't use the same data
    d. All of the above

A lock just provides a way for a program to check if another processor has indicated it wants to use a piece of data. It is up to the program to respect this indication to avoid having two programs touch the same data at the same time.

28. What can happen if people don't use locks correctly in a program?
`2016-Aug-Parallelism-2`
    a. Program crashes
    b. Incorrect results
    c. Death
    d. All of the above

The program can easily generate the wrong values and/or crash, which can (and has!) lead to deaths.

29. The fastest computer in the world, the Chinese Sunway TaihuLight supercomputer, has 10.7M cores. What would be the speedup of a program that was 99.99% parallel?
`2016-Aug-Parallelism-3`
    a. 1,70x
    b. 10,000x
    c. 1,000,000x
    d. 10,700,000x

    From Ahmdal's law s = 1/(1-0.9999+0.9999/10.7M)=9,999.

30. The fastest computer in Sweden, Beskow at KTH, has 53,632 processor cores. What would be the speedup of a program that was 99.99% parallel?
`2016-Aug-Parallelism-4`
    a. 5x
    b. 536x
    c. 8,400x
    d. 53,632x

    From Ahmdal's law s = 1/(1-0.9999+0.999/53632)=8,429.

31. Which was not an example of using parallelism to improve performance?
`2016-Aug-Parallelism-5`
    a. Double-pumped register file access
    b. Pipelining
    c. VIPT
    d. Moving the branch calculation earlier

    Double-pumped register file access allows two accesses in the same cycle, pipelining executes 5 instructions at the same time, and VIPT searches both the TLB and the cache at the same time. Moving the branch calculation earlier does improves performance by not wasting slots in the pipeline, but not by doing something in parallel.

32. The cache for processor 2 wants to load an address A. What needs to be true about the data in processor 1's cache such that processor 1 must send its data to processor 2 to avoid processor 2 getting incorrect data
`2016-Aug-Parallelism-6`
    a. It has the line (tag match and valid)
    b. It has the line (tag match and valid) and it is not dirty
    c. It has the line (tag match and valid) and it is dirty
    d. It has the dirty line (tag and dirty)

    If processor 1 has the line (the tag matches) and the line is valid (it's really there) and it is dirty (it has been changed) then that means the most recent value is in processor 1 and it has to send it to processor 2. If the line is not there (no tag match or not valid) or it is not dirty, then processor 2 can use a copy in its own cache or the one in memory.

33. Which of the following statements is not true about locks?
`2014-Dec-Parallelism-1`
    a. Locks do not prevent any other processor from writing the data while it is locked
    b. Locks are special variables administrated by the operating system
    c. Locks are a signaling mechanism that let the programs keep track of whether any other processors are currently accessing the data
    d. One lock per piece of shared data is needed to synchronize processes properly

    Locks are just normal variables that the program can use to synchronize properly.

34. Which of the following problems does cache snooping solve?
`2014-Dec-Parallelism-2`
    a. Keeps only the necessary caches up-to-date with changes that other processors make in their own caches.
    b. Keeps all the caches up-to-date with changes that other processors make in their own caches
    c. Protects data so that it is not accessed from multiple processors at the same time.
    d. None of the above

    Snooping only makes sure that changes made in a processor's cache are seen in the other caches as well.

35. Why did manufacturers start making multicore processors?
2014-Dec-Parallelism-3
    a. They can't increase the clock speed due to power and heat
    b. They can't build circuits that run faster due to having too small transistors
    c. It is easier to write software for parallel processors than for a single fast processor
    d. All of the above

It's power and heat. The transistors stopped scaling in voltage, so the power does not go down as they get smaller. Running them faster uses too much energy and generates too much heat. We can build faster circuits, though. It is much easier to write software for a single processor than for parallel processors because you don't have to split up work and synchronize.

36. The fastest computer in the world, the Chinese Tianhe-2 supercomputer, has 3.12M processor cores. What would be the speedup of running a program that was 99.999% parallel on this machine?
2014-Dec-Parallelism-4
    a. 35,000x faster
    b. 97,000x faster
    c. 100,000x faster
    d. 3,120,000x faster

99.999% takes 1/53632 of the time plus we have 0.001% that doesn't speed up.
0.99999/53632+0.00001=0.0000186+0.00001= 0.00002864539827. 1/ 0.00002864539827=34,909x.

37. The fastest computer in Sweden, Beskow at KTH, has 53,632 processor cores. What would be the speedup of running a program that was 99.999% parallel on this machine? (Note that this machine has 1/60th as many cores as the Tianhe-2. For fun, consider which machine is better in terms of performance/dollar?)
2014-Dec-Parallelism-5
    a. 35,000x faster
    b. 97,000x faster
    c. 100,000x faster
    d. 3,120,000x faster

99.999% takes 1/53632 of the time plus we have 0.001% that doesn't speed up.
0.99999/53632+0.00001=0.0000186+0.00001= 0.00002864539827. 1/ 0.00002864539827=34,909x.

38. My laptop has 2 CPU processor cores that run at 2GHz (billion cycles per second) and can each do 4 instructions per cycle and a GPU (graphics processor) with 16 processor cores that run at 1GHz and each do 2 instructions per cycle. You can use either the CPU or the GPU at any given time. What is the fastest rate I can run a program that is 50% parallelizable in GIPS (giga (billion) instructions per second)?
2014-Dec-Parallelism-6
    a. 12 GIPS
    b. 17 GIPS
    c. 20 GIPS
    d. 32 GIPS

Running the serial part on one CPU core gives us 50% at 4*2GIPS and the parallel part on the GPU gives us 50% at 16*2*1GIPS = 20 GIPS total. All on the GPU would give us 50% at 2*1GIPS and 50% at 16*2*1GIPS = 17 GIPS. All on the CPU would give us 50% at 4*2GIPS and 50% at 2*4*2GIPS = 12 GIPS.

39. Why do we do parallelism?
2015-Apr-Parallelism-1
    a. To reduce power
    b. To get better performance
    c. To save power
    d. To get better performance, save power and overcome Moore's law

Parallelism can be more power-efficient, but it doesn't in itself reduce power. We only do it for performance. We wouldn't do parallelism if it helped overcoming Moore's Law but not improving performance.

40. What is a lock?
2015-Apr-Parallelism-2
    a. It is a mechanism to protect data so only one processor can access it at a time
    b. It is a mechanism to let a program check if another processor is using the data

c. It is a mechanism to synchronize data so that different caches see the same value
d. All of the above

All locks do is provide a way for a program to check if another processor doing something. They don't actually fix any synchronization, but they allow you to write a program that is synchronized.

41. How much faster can my program run if I have 9000 cores and 20% of the program cannot be parallelized?
`2015-Apr-Parallelism-3`
    a. 5x
    b. 1800x
    c. 7200x
    d. 9000x

The best we can do is to make the 80% take zero time, which would be 5x faster.

42. Which of the following used parallelism to improve performance?
`2015-Apr-Parallelism-4`
    a. Associative caches
    b. Address and branch calculations
    c. Register file access
    d. All of the above

Caches use parallel address comparisons and lookups; the pipeline does the address and branch comparison in parallel with different ALUs; and the register file supports reading and writing 3 registers in parallel.

43. The Intel Haswell Processor (CPU) with the Iris Pro 5100 integrated Graphics Processor (GPU) has the following characteristics: 40 GPU cores that can each execute 20.8B floating point operations per second per core and 4 CPU cores that can each execute 56B floating point operations per second per core. Which is better for a program that is 50% parallel?
`2015-Apr-Parallelism-5`
    a. Run on the CPU cores
    b. Run on the GPU cores
    c. Same
    d. Need more information

Consider how long a program with 1000B instructions will take to run. On the GPU, it will take 500B/(40*20.8)+500B/(1*20.8)=24.6 seconds, while on the CPU it will take 500B/(4*46)+500B/(1*56)=11.16 seconds. Even though the GPU is much more parallel, 50% needs to run on a serial processor so the CPU wins.

44. The Intel Haswell Processor (CPU) with the Iris Pro 5100 integrated Graphics Processor (GPU) has the following characteristics: 40 GPU cores that can each execute 20.8B floating point operations per second per core and 4 CPU cores that can each execute 56B floating point operations per second per core. How much faster is running the parallel part on the GPU and the serial part on the CPU than running the whole thing on just the GPU?
`2015-Apr-Parallelism-6`
    a. 4% faster
    b. 1.4x faster
    c. 2.8x faster
    d. Need more information

The whole thing on the GPU is 50% at 40*20.8B plus 50% at 20.8B, for an average of 426B instructions per cycle. Putting the serial part on the CPU gives us 50% at 40*20.8B plus 50% at 56B, or 444B, which is 4% faster.

45. Why can't we just keep increasing frequency to get better performance?
`2013-Jan-Parallelism-1`
    a. Increasing frequency increases power
    b. We can't afford to cool chips beyond about 300W
    c. Voltage isn't going down so power goes up
    d. All of the above

46. If I have a Intel Xeon Phi processor with 62 cores, what is the best speedup I could get from a program that is 99% parallelizable?
`2013-Jan-Parallelism-2`
    a. 38.5
    b. 58.4

c. 61.4
d. 62.0

47. What do you not need to add to the pipeline to support executing load/store instructions in parallel with other instructions?
2013-Jan-Parallelism-3
    a. Extra ALU
    b. Extra data memory ports
    c. Extra register file ports
    d. Extra instruction memory ports

48. Which of the following does not use parallelism to improve performance?
2013-Jan-Parallelism-4
    a. Associative caches
    b. Register File
    c. ALUs
    d. None: they all use parallelism

49. What does adding instruction level parallelism to the processor not do?
2013-Jan-Parallelism-5
    a. Make the processor more complicated
    b. Reduce hazards
    c. Reduce CPI
    d. Simplify the compiler's job

50. The cache for processor 2 wants to load an address. What does the cache for processor 1 need to check to see if it needs to write back data and tell processor 2's cache to wait for it?
2013-Jan-Parallelism-6
    a. If it has the line (tag and valid) and it is dirty
    b. If it has the line (tag and valid) and it is not dirty
    c. If it has the line (tag and valid)
    d. If it has the dirty line (tag and dirty)

51. Why do we do parallelism?
2015-Aug-Parallelism-1
    a. Faster
    b. Easier
    c. Lower power
    d. All of the above

    Parallel code and hardware is (a lot) harder than serial code and hardware. Parallel execution can be more power-efficient than serial, but it doesn't in itself reduce power. We only put up with the pain and complexity of parallelism for performance.

52. How does a lock help?
2015-Aug-Parallelism-2
    a. Prevents other processors from changing data
    b. Makes sure other processors see changes to data in my cache
    c. Tells other processors that bother to check that they should not change data
    d. Make the code parallel

    The lock serves as a flag that other processors can check to see if they have permission to access some data. It does not stop them from accessing the data if they do not bother to check the flag. The data updates in the cache are handled by the hardware coherency and locks do not make code parallel.

53. What happened in 2003/2004 that led us to have multicore chips everywhere today?
2015-Aug-Parallelism-3
    a. Processors got small enough that you could put multiple cores on the same chip to get better performance
    b. Processors got so hot that it was too expensive to cool them, so manufacturers switched to putting multiple slower (cooler) processors on a chip
    c. Manufacturers couldn't increase the clock speed, so they put multiple cores on the same chip to get better throughput
    d. All of the above

Processors never really got smaller: people kept adding more and more features so they were getting larger even faster than the rate at which transistors were getting smaller. Manufacturers could indeed increase the clock speed (Intel did). The killer was power: it was too expensive to cool faster chips so they switched to multicore and we've been suffering ever since.

54. A program takes 100s to run on a single CPU and 50% of it can be run in parallel. A multicore CPU has 4 processors. How long will it take to run on the multicore CPU?
`2015-Aug-Parallelism-4`
    a. 25s
    b. 37.5s
    c. 50s
    d. 62.5s

50s will still take 50s. But the other 50s will go 4x faster, or take 12.5s. So the program will take 62.5s to finish.

55. A program takes 100s to run on a single CPU and 50% of it can be run in parallel. A graphics processor (GPU) has 240 processors that are each 1/4 as fast as a single CPU. How long will it take to run on the GPU?
`2015-Aug-Parallelism-5`
    a. 0.83s
    b. 50.83s
    c. 200s
    d. 200.83s

50s will take 4x as long to finish (single-core on the GPU is 4x slower). So 50s->200s. The remaining 50s will be parallel, so it will take 50s/(240/4)= 0.83s. So the total will be 200.83s

56. A program takes 100s to run on a single CPU and 50% of it can be run in parallel. A multicore CPU has 4 processors. A graphics processor (GPU) has 240 processors that are each 1/4 as fast as a single CPU. How long will the program take to run if you put the serial part on the best processor and the parallel part on the best processor?
`2015-Aug-Parallelism-6`
    a. 0.83s
    b. 1.66s
    c. 25.83s
    d. 50.83s

50s will take 50s on the CPU. The other parallel part will take 50s/(240/4)=0.83s, so the total will be 50.83s.

57. If we have a program that is 80% parallel and I have 1000 cores, how much faster can we make the program?
`2014-Sep-Parallelism-1`
    a. 1000x
    b. 800x
    c. 5x
    d. 2.5x

We can make the 80% take virtually zero time, but we've still got the 20% that isn't parallel left. So we can run it in 20% of the time, or only 5 times faster.

58. What do you *need* to add to the pipeline to support executing load/store instructions in parallel with other instructions?
`2014-Sep-Parallelism-2`
    a. Extra data memory
    b. Extra data memory ports
    c. Extra register file ports
    d. All of the above

We don't need to add extra data memory or data memory ports. We only need to add ports to read from the register files to read from/write to.

59. Which of the following statement is *incorrect* about locks and a particular piece of data?
`2014-Sep-Parallelism-3`
    a. Locks don't prevent any other processor from changing the data while it is locked.
    b. Locks don't prevent any other processor from accessing the data while it is locked.
    c. Locks enable each processor to safely update the data at the same time.
    d. Locks enable the program to keep track of when other processors are access the data.

Locks only serve to indicate what is going on; they don't allow multiple processors to use the data at the same time without problems.

60. What could happen if we do *not* use atomic instructions while implementing locks?
`2014-Sep-Parallelism-4`
    a. Hazards in the pipeline.
    b. The appearance of shared data in the cache that shouldn't be there.
    c. The data could be changed during the process of acquiring the lock.
    d. A segmentation fault.

Atomic instructions guarantee that the lock will be checked and acquired at the same time without any other processor getting in between the check and acquire and changing the lock or data.

61. A processor can do any three ALU operations (add/sub/etc) at the same time if the instructions are independent. A program has 60% loads/stores and 40% ALU operations. 15% of the instructions depend on a previous instruction and can only execute one at a time. What is the best IPC you could expect?
`2014-Sep-Parallelism-5`
    a. 2.7
    b. 2.1
    c. 1.7
    d. 1.5

Both load/store and ALU operations require the ALU, so 100% of our instructions could be scheduled three at a time. However, 15% have dependencies, so 85% of the time we can find three operations to do on our three ALUs. IPC = 3*0.85 + 1*0.15 = 2.7

62. An Intel Xeon Sandy Bridge processor has 4 processors (cores) and 32kB of L1 cache for each processor, 256kB of L2 cache for each processor, and 20MB of L3 cache shared between all 4 processors. Intel is considering changing the processor to reduce the L3 cache to 13MB, keep the L2 the same, and increase the L1 cache to 64kB. What would you expect to happen when you run 4 programs that each use 3MB of data on the new design compared to the Sandy Bridge? (Assume the only difference is the cache sizes.)
`2014-Sep-Parallelism-6`
    a. No difference
    b. The programs will run slower on the new design because the L3 cache will be almost full when executing, while on the Sandy Bridge there will be plenty of extra space.
    c. The programs will run slower on the Sandy Bridge because it is an older generation.
    d. The programs will run faster on the new processor because the L1 size is bigger

Since all the data of all programs with in L3 cache, the only benefit will be from L1 size that is bigger on the new processor, so programs will run faster on the new machine. It doesn't matter at all that the L3 cache has plenty of space on the Sandy Bridge since the programs aren't using it.

63. Why do we increase parallelism in processors?
`2016-Oct-Parallelism-1`
    a. It's easier to program
    b. It is easier to get performance from parallelism than by increasing the clock speed
    c. It doesn't require changing anything in the processor to get more performance
    d. We're desperate for performance because we can't increase the clock speed due to power

If we could just increase the clock speed of everything that would be a far easier way to get better performance (no changes to the pipeline or program), but unfortunately we can't due to power limits.

64. If we have a program that is 80% parallel and a computer with 1000 processors, how much faster can we make the program?
`2016-Oct-Parallelism-2`
    a. 2.5x
    b. 5x
    c. 800x
    d. 1000x

We can make 80% take virtually zero time (1000x faster for that 80%) but there is still 20% left, so that is only 5x faster.

65. Which technique for improving performance did not involve increasing parallelism?
`2016-Oct-Parallelism-3`
    a. Pipelining
    b. Early branch calculation
    c. Separate instruction and data memories
    d. Increasing the clock speed

Increasing the clock speed allows the existing processor to go faster without needing to have more parallelism.

66. Which of the following statements is incorrect about locks and a particular piece of data?
`2016-Oct-Parallelism-4`
    a. Locks enable the program to keep track of when other processors are accessing the data.
    b. Locks enable each processor to safely update the data at the same time.
    c. Locks don't prevent any other processor from accessing the data while it is locked.
    d. Locks don't prevent any other processor from changing the data while it is locked.

Locks do not allow two processors to update the data at the same time. They only provide a way for processors to indicate to each other that they want or have permission to access a piece of data.

67. What do modern processors have to be able to do to execute more than 1 instruction per cycle?
`2016-Oct-Parallelism-5`
    a. Find independent instructions
    b. Have multiple functional units (e.g., ALUs) to execute multiple instructions at the same time
    c. Load multiple instructions at the same time
    d. All of the above

To get an IPC of 4, a processor needs to be able to find 4 independent instructions at the same time (very, very hard), have 4 ALUs to do 4 calculations at once (expensive), and be able to load 4 instructions in each cycle to keep busy.

68. The cache for processor 2 wants to load an address. What does the cache for processor 1 need to check to see if it needs to write back data in its own cache and tell processor 2's cache to wait for it?
`2016-Oct-Parallelism-6`
    a. If it has the dirty line (tag and dirty bit)
    b. If it has the line (tag and valid bit)
    c. If it has the line (tag and valid bit) and it is not dirty
    d. If it has the line (tag and valid bit) and it is dirty

If the other processor has the data (tag and valid) and it is dirty then the first processor will get the wrong value if it uses what is in its cache, so the other processor needs to write it back and tell the first processor.

69. Why don't we just increase the clock speed instead of building parallel processors?
`2016-Jan-Parallelism-1`
    a. We can't build faster circuits
    b. We can't afford the power of faster circuits
    c. It is easier to make software run faster in parallel
    d. All of the above

We can build faster circuits but they use far too much power. Parallel software is much harder to write due to the need to divide up work and synchronize data accesses.

70. Which of the following designs for improving performance does not require multiple PCs?
`2016-Jan-Parallelism-2`
    a. Simultaneous multithreading (issuing instructions from multiple programs into the pipeline at the same time)
    b. Multicore processors (having multiple processors on one chip)
    c. Dual-issue processors (issuing multiple instructions from the same program into the pipeline at the same time)
    d. All of them require multiple PCs

Dual-issue processors issue two instructions each cycle, so the PC simply counts up by 8. The others require multiple PCs to keep track of which instruction in each program is next.

71. How do locks prevent problems with multiple processors accessing data at the same time?

2016-Jan-Parallelism-3

a. They tell the hardware to prevent multiple accesses to the data
b. They tell the hardware to keep the data synchronized across multiple processors
c. They tell the software when it's processor is allowed to access the data
d. They allow the software to coordinate which processor accesses the data

Locks allow programmers to check if another processor has requested access to some data and request access. It is up to the software to both check and then only access the data if it gets access, as the lock itself does not do anything to enforce synchronization.

72. A program processes images in parallel by splitting them up into 1000x1000 pixel pieces and processing as many of them in parallel as it can on the available processors. How long will it take to process a 3000x3000 pixel image on a machine has 4 processors, each of which can process 1M pixels per second?

2016-Jan-Parallelism-4

a. 2.25 seconds
b. 3 seconds
c. 4 seconds
d. 9 seconds

There are 9 pieces to process. The first 8 will be processed in parallel on 4 processors at the same time in 2 seconds. (4 processors * 2 seconds = 8M pixels). The last 1M will run alone (since there is no more work to do) on 1 processor and take 1 more second, for 4 seconds total..

73. The average smart phone runs 1.7 programs at the same time. (This was true in 2014, at any rate.) What speedup would you expect from switching to a new phone with 8 cores over your old phone with 2 cores, assuming the cores are the same on both?

2016-Jan-Parallelism-5

a. 1x
b. 1.7x
c. 4x
d. 6.8x

Two cores is already enough to get all the benefits of running 1.7 programs at the same time, so switching to 8 cores will not speed things up. It will, however, be likely to use more energy.

74. Which of the following was not needed to handle parallel instruction execution in the pipeline?

2016-Jan-Parallelism-6

a. Hazard detection
b. Branch prediction
c. Dirty bits
d. Forwarding

Hazard detection, branch prediction, and forwarding were all needed to address the problem that we had multiple instructions executing at the same time (in parallel) in the pipeline. Dirty bits were added because we needed to know which data should be written back when it was evicted from the cache.

75. Why did clock speeds stop increasing in 2004?

2013-Aug-Parallelism-1

a. We can't make faster transistors
b. The memory was too slow so there was no point
c. We couldn't cool the chips
d. All of the above

We couldn't cool the chips. We can make faster transistors, and caches help with the memory problem.

76. If a program is 80% parallelizable and I have 5000 processors, what is the best speedup I could achieve?

2013-Aug-Parallelism-2

a. 5x
b. 4000x
c. 5000x
d. Need more information

You can make 80% of the program go 5000x faster. That part is therefore essentially instant, so you are left with the

remaining 20%, or 1/5. That means you are 5x faster.

77. How do locks avoid data synchronization problems?
`2013-Aug-Parallelism-3`
    a. They prevent any other processor from changing the data while it is locked.
    b. They prevent any other processor from accessing the data while it is locked.
    c. They let the program keep track of whether any other processors are currently accessing the data.
    d. They make copies of the data for each processor so they can update them at the same time.

The locks don't actually prevent anything from happening. All they do is let the program know if another processor is trying to access the data. It is up to the program to be well-behaved and avoid accessing the data if another processor is.

78. What problem does cache snooping solve?
`2013-Aug-Parallelism-4`
    a. Prevents multiple processors from accessing the same data at the same time
    b. Keeps DRAM up-to-date with changes that processors make in their own caches
    c. Keeps each processor's cache up-to-date with changes that other processors make in their own caches
    d. All of the above

Snooping only makes sure that changes made in a processor's cache are seen in the other caches as well.

79. What has happened as a result of incorrectly synchronized data accesses from parallel processors?
`2013-Aug-Parallelism-5`
    a. Programs crashed
    b. Banks lost money
    c. People died
    d. All of the above

All of the above.

80. What speedup would we expect from a triple-issue pipeline that can do 1 load/store instruction and 2 other instructions at the same time for a program that is 1/3 load/store instructions?
`2013-Aug-Parallelism-6`
    a. Less than 3x
    b. 3x
    c. More than 3x, but less than 6x
    d. More than 6x

Less than 3x. We know that programs have all sorts of nasty data and control dependencies that mean that even though 1 out of 3 instructions is a load/store (e.g., there are 2 non-load store instructions and 1 load/store for every three instructions) we won't always be able to execute them at the same time. (Remember that we can't forward from instructions that haven't executed yet.)

81. Why do we do parallelism?
`2013-Apr-Parallelism-1`
    a. Easier
    b. Faster
    c. Fewer transistors
    d. All of the above

Parallelism is a lot harder to do and requires more transistors, but it can make things run faster.

82. Why don't we just increase the clock speed?
`2013-Apr-Parallelism-2`
    a. Can't build faster circuits
    b. Can't afford more power
    c. Can't make memory faster
    d. All of the above

We can build faster circuits, but they'd use up a lot more power. The speed of memory doesn't prevent us from increasing clock speed, but it does make it less beneficial.

83. What is the difficult part about parallelizing summing up 100M numbers to get no wasted (idle) time?
`2013-Apr-Parallelism-3`
    a. Dividing up the work to have the same amount for each processor
    b. Dealing with data that multiple processors need to share
    c. Synchronizing when the processors are done with their portions
    d. All of the above

Dividing the numbers up is easy (just give each processor 1/n) and there is no data that overlaps. The hard part is when each processor is done you have to synchronize to add up the final results.

84. How much faster can my program run if I have 2000 cores and 10% of the program cannot be parallelized?
`2013-Apr-Parallelism-4`
    a. 10x
    b. 90x
    c. 200x
    d. 2000x

The best we can do is to make the 90% take zero time, which would be 10x faster.

85. How do locks help with parallel programs?
`2013-Apr-Parallelism-5`
    a. The protect a memory location so only one processor can change it at once
    b. They make sure data is updated across all processors
    c. They serve as an indication that another processor is changing the data
    d. They force the cache to keep the data synchronized

Locks are just an agreed upon indicator that someone else is accessing the data. We need to check the lock and not access the data if someone else has it for them to work.

86. A processor can do two ALU operations (add/sub/etc.) at the same time for two instructions that come one after the other. A program has 40% load/stores, and 60% ALU operations. 10% of the instructions depend on a subsequent instruction. What is the best IPC you would expect?
`2013-Apr-Parallelism-6`
    a. 1.5
    b. 1.6
    c. 1.8
    d. 1.9

Both load/store and ALU operations require the ALU, so 90% of the time we can find two instructions to do on our two ALUs, or 2*0.9+1*0.1=1.9.

Make sure you circle the answers on the answer page.
Answers on other pages will NOT be graded.