Computation I (5EIA0) Exam Training (CA) 2021

Original authors SPIM: Tom van Nunen, Sjors van Riel, Thomas Lippens
Version 2021-2022: Esther Maas

Monday November 1, 2021

1 CA questions

Question 1

We consider a program P in which *lw* and *sw* instructions together are 40% of all instructions.
Processor A with a cache has the following metrics during execution of program P:

- D-cache miss rate = 20%
- I-cache miss rate = 8%
- Miss penalty = 50 cycles

D-cache misses occur only during load and store instructions. Also given is an ideal processor X that has no cache misses for program P with base CPI = 4.0.

How much is the ideal processor faster than our processor A?

Question 2

A data cache is indexed with a 10-bits index, tag size is 18 bits. Each cache block contains a tag, a block of words, 1 dirty bit, 1 valid bit; each word contains 4 bytes. The address size is 32 bits, and the memory is byte addressable.

The cache is four-way associative.

What is the total cache size (in bits)?

Question 3

A data cache is indexed with a 12-bits index. Each cache entry contains 8 words, a tag, 1 dirty bit and 1 valid bit; each word contains 4 bytes. The address size is 32 bits, and the memory is byte addressable.

Calculate the size of a tag (in bits).

Question 4

Consider an instruction cache. How many MIPS *sw* instructions can be stored in one cache block of 16 bytes?

Question 5

Consider a cache which is able to store 256 bytes of memory data. To what cache block (entry) does the **memory address 516** map, if the cache has 1-word block size and is direct mapped?

Question 6

Consider a pipelined MIPS implementation with 5 stages as shown below. Assume that a test program enables ideal pipelining with no hazards.

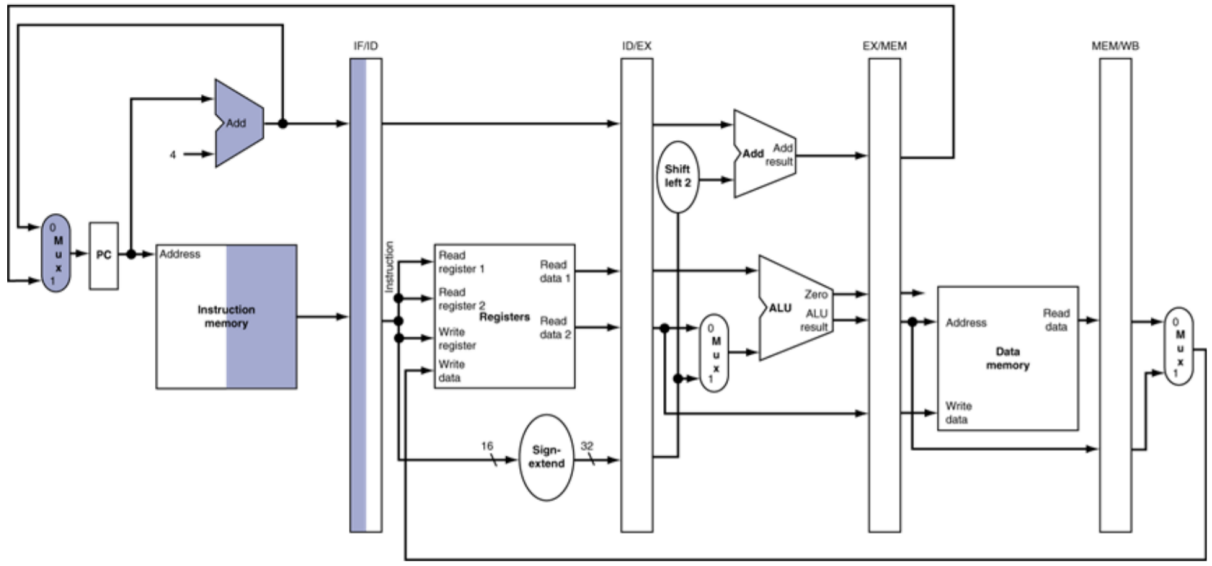


Figure 1:

Assume the following (theoretical) timings for hardware blocks, all in ns:

Adder: 2,
ALU: 4,
Register read: 2,
Register write: 2,
Instruction memory access: 2,
Data memory access: 2,
Each mux: 1,
Sign-extend: 0,
PC: 0,
Shiftleft 2: 0

What is the minimum number of instructions per second (in MHz) supported by this processor?

Question 7

Consider a single-cycle (non-pipelined) MIPS implementation as shown below:

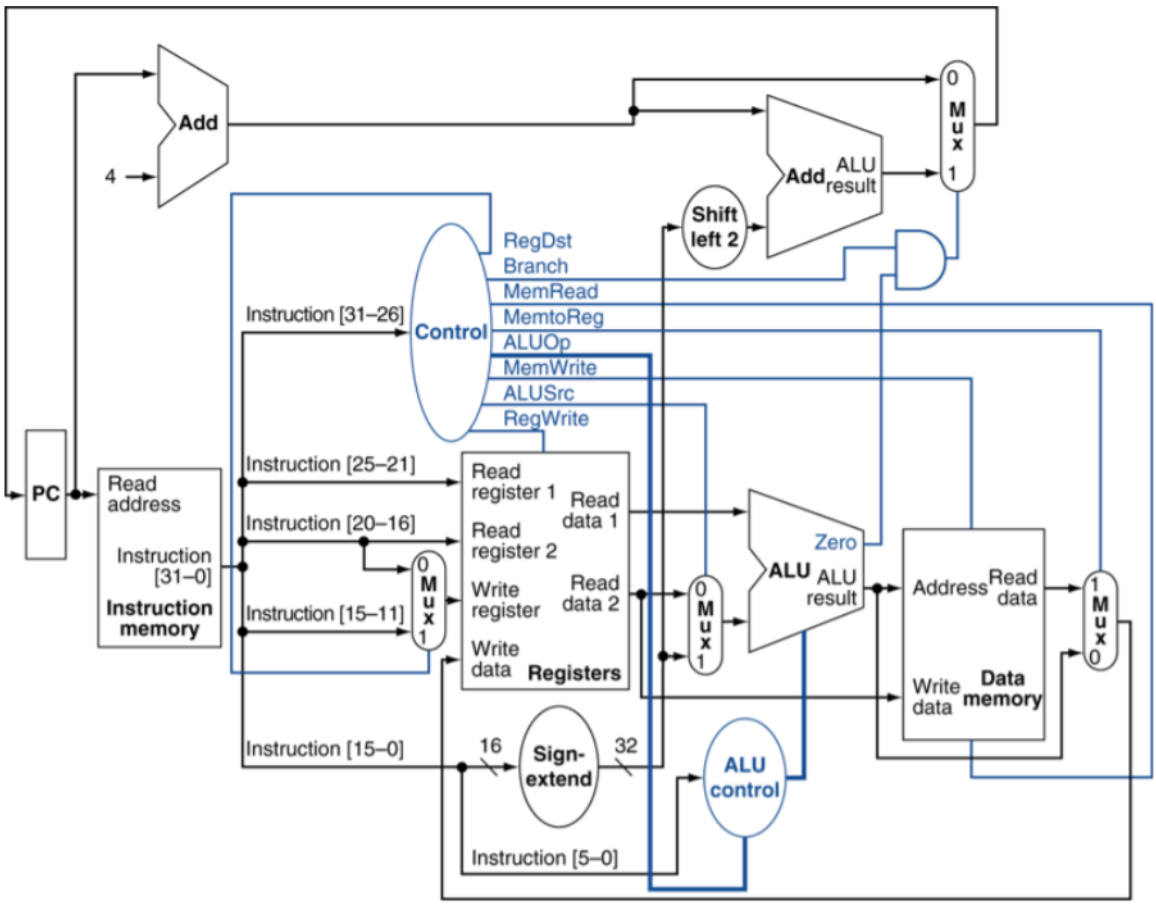


Figure 2:

Assume the following timings for hardware blocks, all in ns:
Adders: 1,
ALU: 6,
Register read: 2,
Register write: 2,
Each Instruction memory acces and Data memory acces: 6,
Each Mux: 1,
PC: 0,
Sign-Extend: 0,
Control blob: 1

What is the minimum number of instructions per second (in MHz) supported by this processor?

Question 8

Which hazard type(s) does the *beq* instruction introduce for a pipelined processor in the following instruction sequence:

```
add $s1 $s2 $s3
beq $s1 $s4 Label1
some instruction
some instruction
Label1: some instruction
```

select one:

- The instruction does not introduce any hazards%
- Control hazard only%
- Data and Control hazards%
- Structural and Control hazards
- Data hazard only

Question 9

Imagine our pipelined processor (diagram below) is executing the following piece of a program:

sub $s0 $s2 $s3
add $s4 $s5 $s0

Instruction execution stages of the pipelined processor:

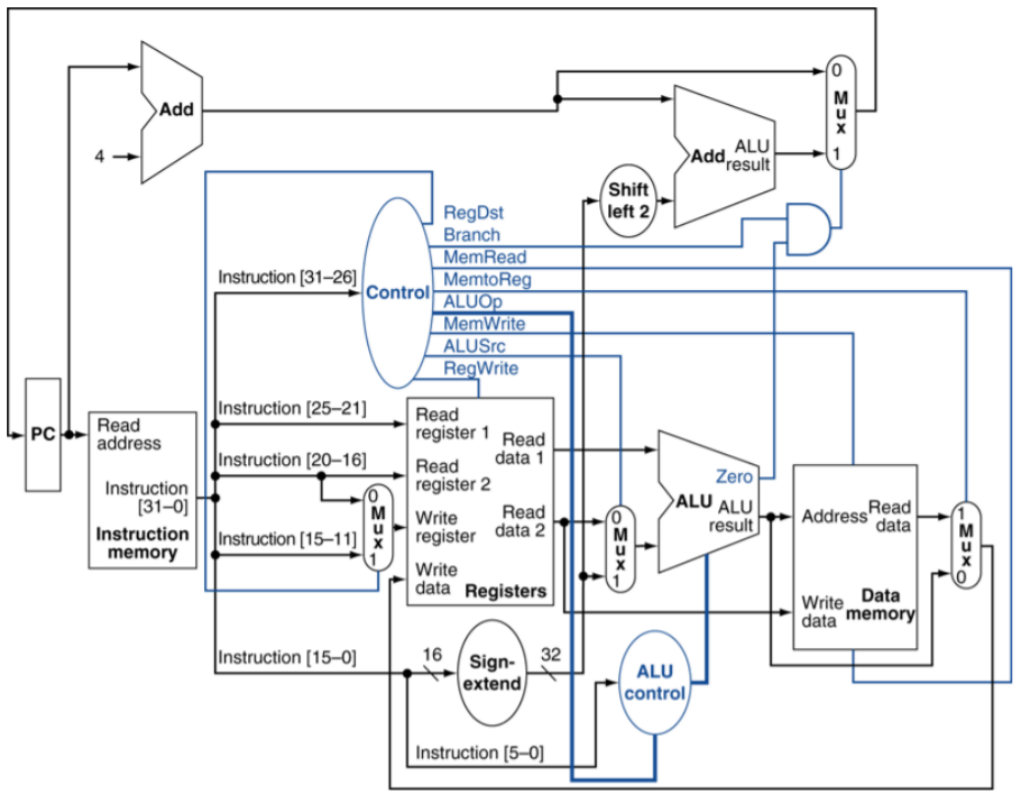


Figure 3:

Each stage (timeframe) takes 20 nanoseconds.
No hardware tricks or bypassing are deployed.
The RegisterWrite resource does not provide the updated register value to the RegisterRead resource at the same stage (timeframe). In other words, the updated register value can be read only in the next timeframe.
What will be the duration (in nanoseconds) of the processor stall due to the pipeline hazard in the program above?

Question 10

What will be the memory address (in decimal number) of the next instruction to execute after the following instruction:

0x08000001

This instruction above is given in hexadecimal format and is located in the address 0x00fffffff

Question 11

Given the folowing C code:

```
f = A[0] - B[1] - C[3];
```

Assume that the base addresses of arrays A, B and C are already loaded to the registers $s4, $s5, $s6. Convert the C code into the most efficient MIPS assembly code. What is the minimum number of assembly instructions that are required to implement this C code?

Question 12

The *loop* label is located 4 instructions above the beq instruction:

```
Loop:
some instruction
some instruction
some instruction
beq $t0,$t2,Loop
```

What is the 32-bit binary machine instruction for the beq instruction?
Do not use spaces in you answer.

Question 13

Which number does the following SPIM code print out, if you enter 4 upon the console user-input request? Enter -1 if the program does not print out anything.

```
main:
    move $22,$0
    li $21, 50
    li $v0, 5
    syscall
    move $23,$v0
loop:
    beq $0,$23,quit
    add $22,$22,$21
    addi $23,$23,-1
    j loop
quit:
    move $a0, $22
    li $v0, 1
    syscall
```

Question 14

What does this program perform, if you enter 10 upon every user-input request?

```
    .text
main:
    li $v0, 5
    syscall
    move $t0, $v0
    li $v0, 5
    syscall
    move $t1, $v0
    sub $t2, $t0, $t1
    move $a0, $t2
    li $v0, 1
    syscall
    li $v0, 10
    syscall
```

a The program will print 10 to console

b The program will load 10 into register

c The program will load 20 into register

d The program will print 20 to console

e The program will load 0 into memory

f The program will print 0 to console

g The program does something different than listed in the other options.

Question 15

Consider the following processor design.

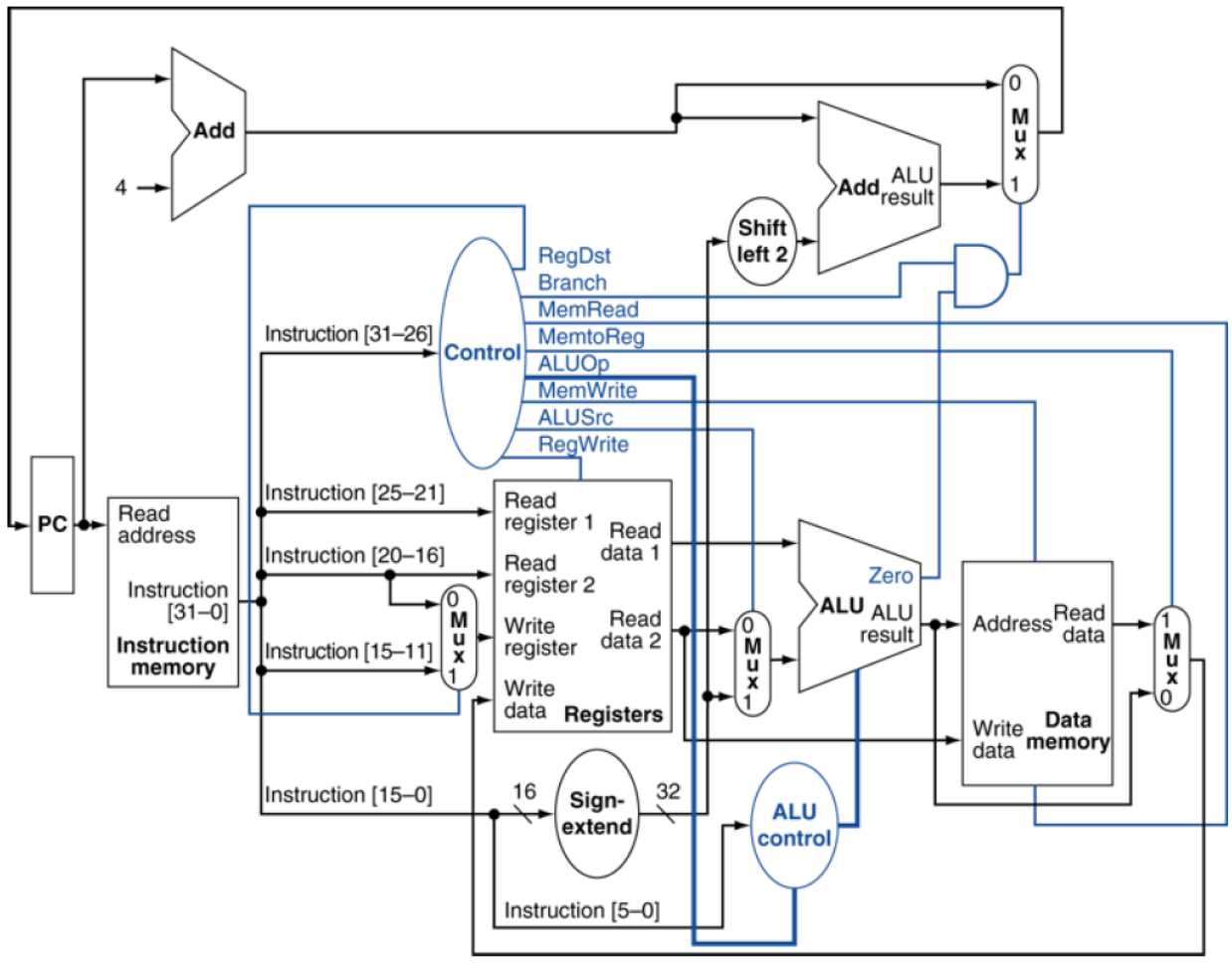


Figure 4:

Which resources are used in execution of the following instruction:

```
lw $t0, 16($s1)
```

The PC update may be ignored.

a Instruction Memory, Mux, Register Read, Sign-extend, Mux, ALU, Data Memory

b Instruction Memory, Register Read, Sign-extend, Mux, Data Memory, Mux, Register Write

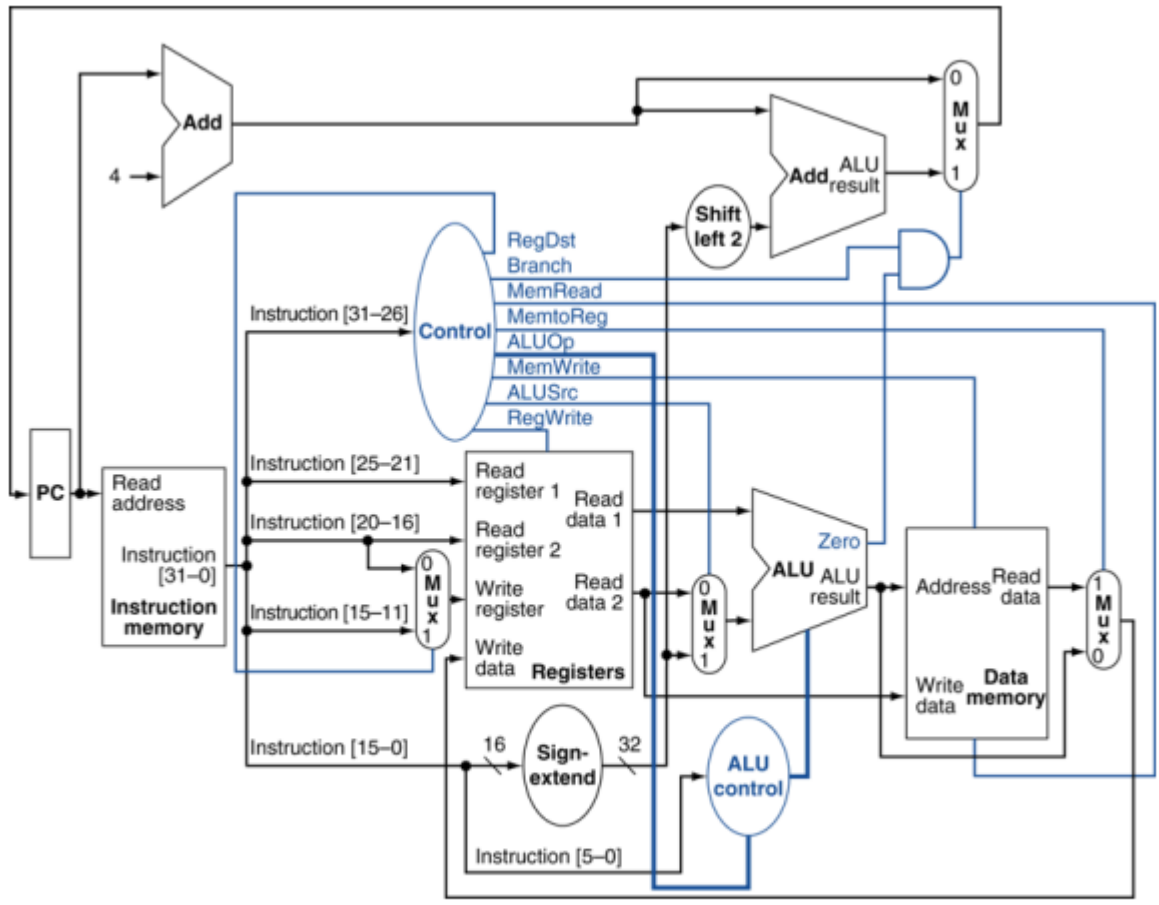
c Instruction Memory, Mux, Register Read, Sign-extend, Mux, Data Memory, Mux, Register Write

d Instruction Memory, Mux, Register Read, Sign-extend, Mux, ALU, Data Memory, Mux, Register Write

e Instruction Memory, Register Read, Sign-extend, Mux, ALU, Data Memory, Mux, Register Write

Question 16

Consider the processor design given below.



Which resources are included in the datapath of the following instruction:

add $s0, $s1, $s2

The PC update may be ignored.

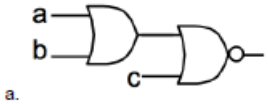
Figure 5:

1. a. Instruction Memory, Mux, Register Read, Mux, ALU, Data Memory, Mux, Register Write
2. b. Instruction Memory, Register Read, Mux, ALU, Mux, Register Write
3. c. Instruction Memory, Mux, Register Read, Mux, ALU, Mux, Register Write
4. d. Instruction Memory, Mux, Register Read, Mux, ALU, Mux, Data Memory

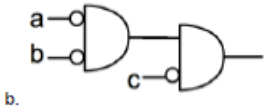
Question 17

To which circuit is the expression NOT(AND(a,b,c,)) logically equivalent?
Select one:

a.



b.



c.

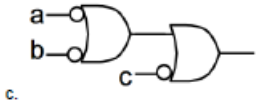


Figure 6:

Question 18

A half adder circuits adds two bits (a, b) and computes the sum (s) of a and b, and the carry (c). This is the truth table of the half adder circuit:

Table 1: Truth table of the half adder

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

This is a gate-level implementation of the half adder, but one logical gate is missing.

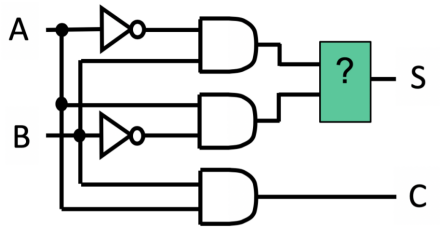


Figure 7:

What is the correct logical gate that should be inserted?

Select one:

1. XOR gate
2. OR Gate
3. XNOR Gate
4. NAND Gate
5. NOR Gate
6. AND Gate
7. NOT Gate

2 Spim

In the SPIM programming exercises the following SPIM directives are used:

`.text`	Start text segment.
`.data`	Start data segment.
`.asciiz str`	Store the string in data memory
`.word 1,40,...,n`	Store `n` words (32 bit) into data memory.i
`.space n`	Reserve `n` bytes in data memory.

This are the SPIM System Calls:

Table 2: SPIM System Calls

Function	Syscall (value in $v0)	Arguments	Return value
Print integer	1	$a0: integer	
Print string	4	$a0: string	
Read integer	5		$v0: integer
Exit	10		

Printing Negative Numbers

Write a SPIM program for printing negative integer elements of an array numbers with length `length`. You can globally allocate the numbers by:

```
numbers:  .words 79, -13, -90, 80, 12, 0, -4, 23, -45, 7
length:  .word 10
```

Calculating Sum of Absolute Values

Extend the program to print the sum of the absolute values in the array instead of all the negative values.