1 ISA 1 2021-03-Retake

[10 points] For the given *clear_array* code below, write the corresponding MIPS assembly code.

C code:

```
void clear_array (int *arr, int n) {
    int i = 0;
    while (i < n) {
        arr[i] = 0;
        i++;
    }
    return;
}
```

C Code explanation:

One variable *i* is initialized with 0.

The *i* variable is the iteration variable that is used to check if we should keep looping the loop and also used to index the array.

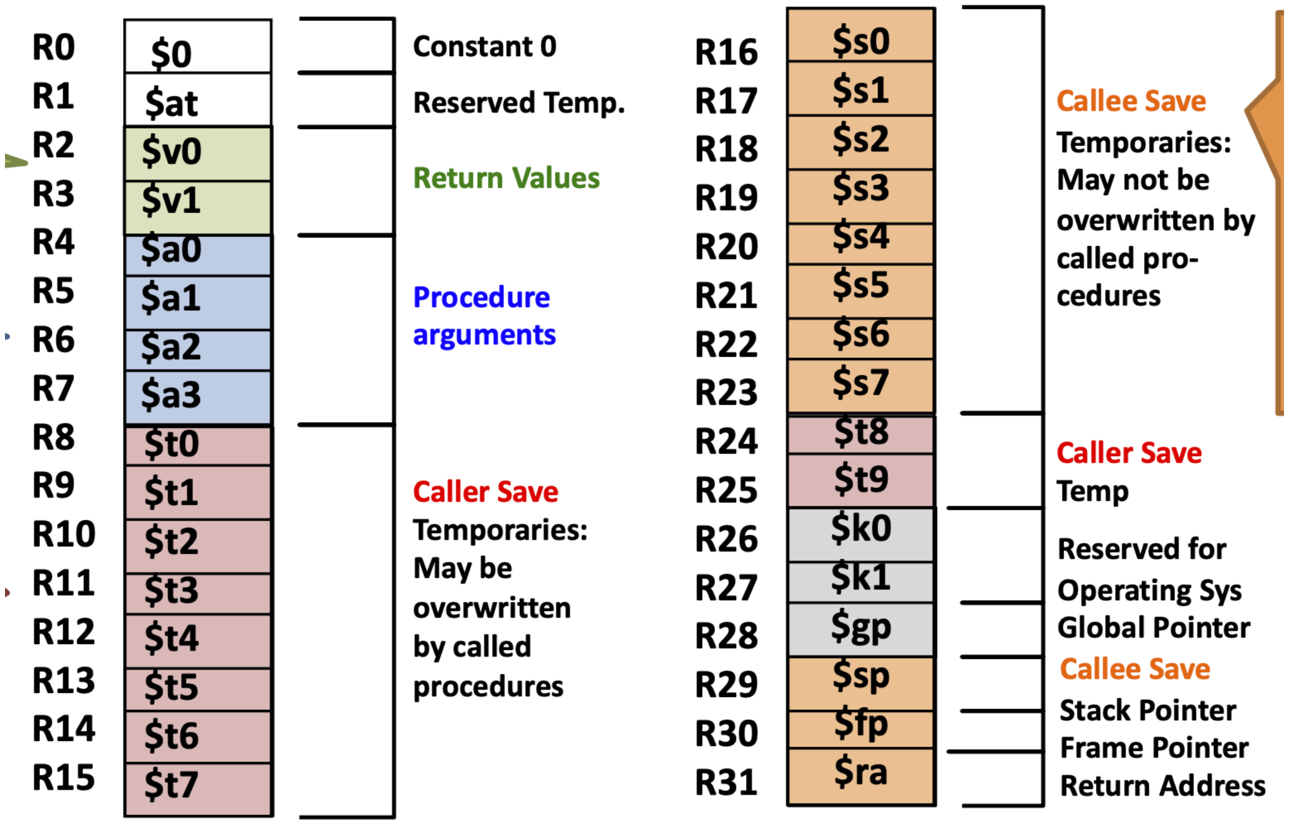
We loop until the *i* variable becomes equal or greater than *n.*

Inside the loop, we clear the array element value to 0. Then we increment the i value.

Once the loop is done, the function returns to the caller.

Register mappings:
$a0: base address of *arr*
$a1: n - the length of *arr* (number of elements of the array)

For any other variable to register mappings used in your assembly code, follow the calling conventions.



Write the MIPS assembly for the *clear_array* function.

Fill your assembly code in here.

Maximum marks: 10

2 ISA 2 2021-03-Retake

[12 points] Assume that you have implemented the *clear_array* function from the prior question. Now you want to call your *clear_array* function from another function called *prepare_array*.

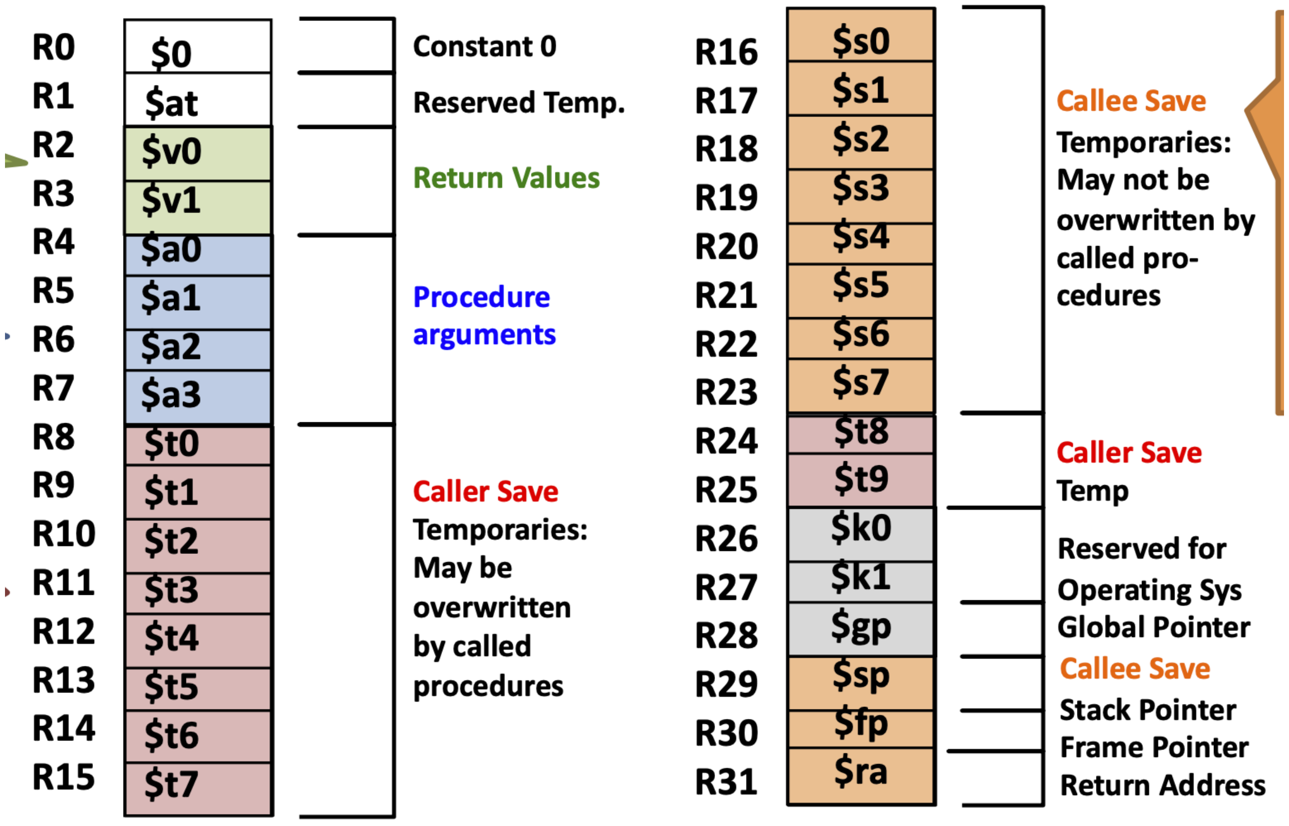
The prepare_array will allocate an array using the call to malloc.
malloc(16*4) will allocate an array of 16 words (4B), and return the base address of the array.

Input & Output of *prepare_array*
INPUT: $a0 -> length of array
OUTPUT: $v0 -> base address of array.

```
prepare_array:

   mov $t0, $a0, $zero
   sll $a0, $a0, 2
   jal malloc

   mov $a0, $v0, $zero
   mov $a1, $t0, $zero
   jal clear_array

   jr $ra
```

Your task:

1. **[4 points]** Identify which registers in use by *prepare_array* are caller save registers and callee save registers, respectively, according to the convention.
2. **[4 points]** Add appropriate instructions to save/restore the registers onto/from the stack, to conform to the MIPS register convention.
3. **[4 points]** Should the $ra register be saved in the 'prepare_array' function? What should it be saved as (caller/callee?) Where should it be saved and where should it be restored?



R0	$0	Constant 0
R1	$at	Reserved Temp.
R2	$v0	Return Values
R3	$v1	
R4	$a0	Procedure arguments
R5	$a1	
R6	$a2	
R7	$a3	
R8	$t0	Caller Save Temporaries: May be overwritten by called procedures
R9	$t1	
R10	$t2	
R11	$t3	
R12	$t4	
R13	$t5	
R14	$t6	
R15	$t7	
R16	$s0	Callee Save Temporaries: May not be overwritten by called pro-cedures
R17	$s1	
R18	$s2	
R19	$s3	
R20	$s4	
R21	$s5	
R22	$s6	
R23	$s7	
R24	$t8	Caller Save Temp
R25	$t9	
R26	$k0	Reserved for Operating Sys
R27	$k1	
R28	$gp	Global Pointer
R29	$sp	Callee Save Stack Pointer
R30	$fp	Frame Pointer
R31	$ra	Return Address

(Reference for register saving convention)

Fill in your answer here

Maximum marks: 8

3 Datapath 2021-03-Retake

[13 points]

[7 points] Firstly, assume that we have a single-cycle processor (not pipelined). Each part of the processor takes to following amount of time:

- Instruction Fetch: 200 ps
- Register Read: 100ps
- ALU operation: 200 ps
- Data access: 200 ps
- Register Write: 100ps
- **Total:** 800 ps

The chip designers decided to add a dedicated adder at the "data access" logic for the address calculation. (Load/store no longer relies on the ALU, and generates address using its own adder). This changed the time of each part of the processor to the following:

- Instruction Fetch: 200 ps
- Register Read: 100ps
- ALU operation: 200 ps
- Data access: **250 ps**
- Register Write: 100ps

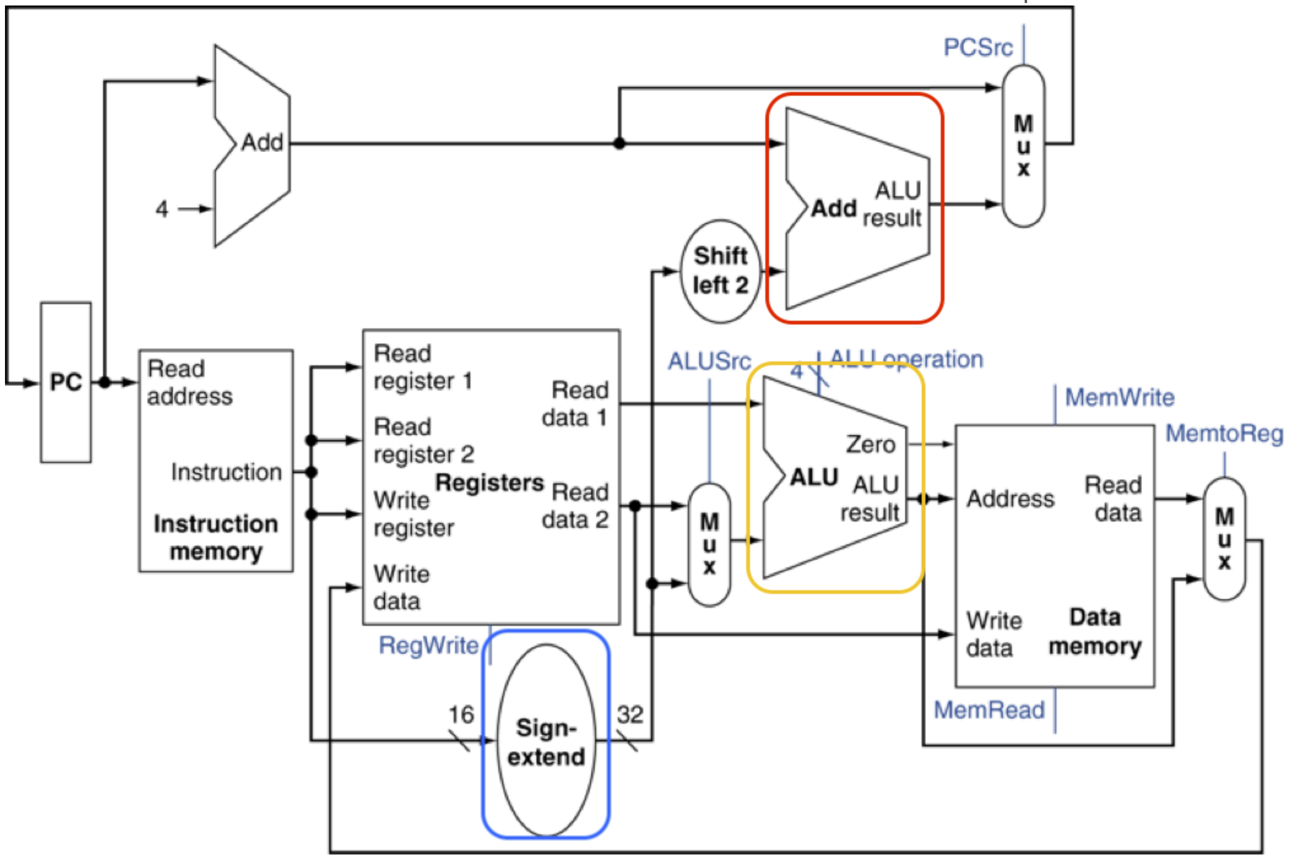
Based on what you have learned about the datapath, answer the following questions.

1. **[3 points]** Calculate the time it takes to do:
    1. add instructions
    2. branch instructions
    3. load instructions
2. **[2 points]** Based on your calculations above, what is the fastest rate at which the processor can run at?
3. **[2 points]** Explain why or why not this optimization of adding a dedicated adder at the "data access" logic is beneficial.

[6 points] Secondly, coming back to the original MIPS datapath design answer the following question:

4. **[3 points]** Why is there an additional adder (red box) to calculate the branch address? Why not use the ALU (Yellow box) to do the branch address calculation?

5. **[3 points]** Explain the purpose of the Sign extender (blue box) on the branch address calculation. (Hint: what would happen if the sign extender did not exist?)



Fill in your answer here

Words: 0

Maximum marks: 13

4 Pipeline 2021-03-Retake

[6 points] A single-cycle processor takes 100ns for each instruction. The processor can be pipelined up to 5 stages, and takes 20ns for each stage. Furthermore, the pipeline register adds another 5ns per stage.

1. **[2 points]** Calculate the latency (time to complete each instruction) of the pipelined processor.
2. **[2 points]** Calculate the throughput (how many instructions complete in a given number of cycles) of a single-cycle processor, and a 5 stage pipelined processor in a 500ns window.
3. **[2 points]** Based on your prior two answers, what is the advantage and disadvantage of the two types of processors (single-cycle vs. pipelined).

Fill in your answer here

Format

Words: 0

Maximum marks: 6

5 Pipeline Hazards 2021-03-Retake

[11 points] Assume the following sequence of instructions, and assume that it is executed on a 5-stage pipelined datapath. Justify your answers with up to four sentences of explanation.
If you want to supplement answers with a drawing, upload drawing on next question!

```
add r1, r2, r3
lw r4, 0(r1)
add r2, r4, r2
add r1, r5, r2
sw r4, 0(r1)
```

1. **[4 points]** Identify the pipeline hazards when running the assembly code above. Identify the pair of instructions that cause the hazards and the type of hazards.
2. **[4 points]** If we implement forwarding, identify the hazards (from the question before) that can be solved by the forwarding. Also identify the hazards that cannot benefit from the forwarding.
3. **[3 points]** Assume the processor does not have logic to identify hazards and install stalls. It is your task to add in the correct number of stalls for any potential hazards that cannot be fixed with forwarding. In the code above, where and how many stalls will you insert into the code? (Assume forwarding has been implemented)

Fill in your answer here

Format

Words: 0

Maximum marks: 11

6 (Optional) Pipeline drawing upload

(Optional) If you want to supplement your pipeline hazards answer with a drawing (hand drawn or drawn with Power point etc.) please upload here.

This is optional. If you don't need drawings, skip this question!

Upload your file here. Maximum one file.

All file types are allowed. Maximum file size is **1 GB**

Select file to upload

Maximum marks: 0

7 Branch Prediction 2021-03-Retake

[8 points] Answer the questions below about branch prediction.

1. **[2 points]** Explain what the branch delay slot is, when it can be used to improve performance.
2. **[2 points]** Describe a case when the branch delay slot cannot be filled, and what must be done in such cases to ensure correct(desired) execution of the code.
3. **[2 points]** For unconditional jump instructions, assume that the target to jump to is decoded at the Instruction Decode stage (second stage in the 5-stage pipeline). Will this cause a control hazard. If so, of how many cycles?
4. **[2 points]** List two or more methods to alleviate the control-hazard caused by unconditional jumps from the prior question.

Fill in your answer here

Format

Words: 0

Maximum marks: 8

8 I/O 2021-03-Retake

[5 points]

1. **[2 points]** Explain the benefits of using a Direct Memory Access (DMA) to transfer data instead of having the CPU move the data using CPU instructions.
2. **[3 points]** Assume that it takes 30 usec(micro seconds) for an OS to receive an interrupt and start processing the received packet. New ultra low latency SSDs take about 30 usec to process a read request. In this case, discuss whether you would use polling or interrupts to handle I/O request completions. (Hint: most network or HDD takes milliseconds. the ultra low latency SSDs are much quicker).

Fill in your answer here

Format

Words: 0

Maximum marks: 5

9 Cache address organization and VIPT caches 2021-03-Retake

[5 points] Cache Organization

- A byte-addressable computing system, with a word length of 4B (bytes), has a main memory of 4 GB. It also has a 4kB direct-mapped data cache, where each cache line contains sixteen words (64B). Fill out how many bits are required for each part of the cache.

Cache Tag: ☐ bit(s)

Cache set index: ☐ bit(s)

Word in cacheline: ☐ bit(s)

Byte in word: ☐ bit(s)

- Assume a 32kB 8-way set-associative cache with a 64B block. How many bits are used as the index bit? ☐ bits.

[3 points] VIPT Caches

- What is the largest size of a direct-mapped cache when using VIPT with a page size of 8kB?: ☐ kB.
- When using a 8-way set associative cache size with a 8kB page size, what would be the largest cache size of the VIPT cache? ☐ kB.
- Assume a 64B cache block, a 32 bit virtual address space, 32 bit physical address space, and a 8kB direct-mapped cache. What is the number of bits required for the tag bits? ☐ bits

Maximum marks: 8

10 Cache 2021-03-Retake

[8 points]

1. **[4 points]** What is the benefit of having a smaller cache block size? What is the disadvantage of having a smaller cache block size?

2. **[4 points]** Give an example of why a direct-mapped cache can result in conflicts. What is the alternative, and how does it overcome the conflicts of the direct-mapped cache (demonstrate that the alternative overcomes the problem the direct-mapped cache had with the same example).

Fill in your answer here

Format

Words: 0

Maximum marks: 8

11 Virtual Memory 2021-03-Retake

1. **[4 points]** The L1 caches in commercial processors are VIPT caches. What is the benefit of using a Virtually-indexed-physically-tagged cache? Compare against a PIPT cache.
2. **[3 points]** What is the benefit of introducing virtual memory? List at least three benefits that VM provides.

Fill in your answer here

Format

Words: 0

Maximum marks: 7

12 Parallelism 2021-03-Retake

[4 points] Amdahl's law helps estimate the potential speed up of a program when using parallel resources to speed up the program execution time. What does Amdahl's law tell us about two programs A: which has a serial region of 80%, and program B: which has a serial region of 20%? (Explain which program can be sped up more, and what the potential speed up of A, and B is).

Fill in your answer here

Maximum marks: 4

13 Arithmetic

Which of the following number formats has only one zero?

- ○ Two's Complement
- ○ Signed Magnitude
- ○ More than one above
- ○ Floating Point

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?

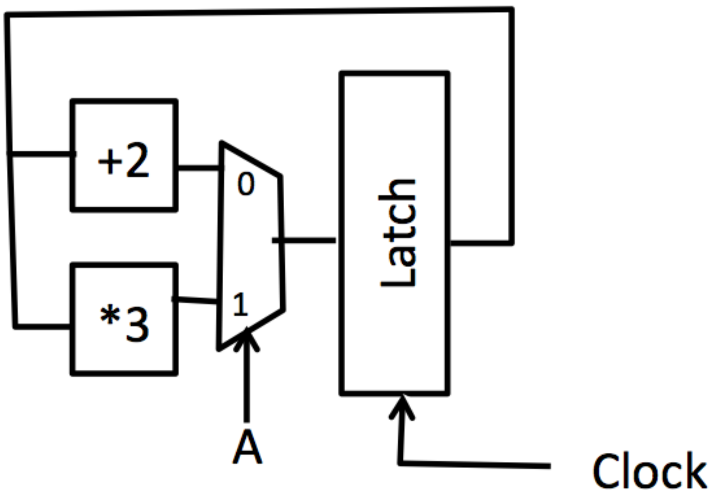
- ○ Signed and unsigned add is the same in hardware, result is always the same
- ○ It reduces the instruction latency
- ○ The output is always the same, but the overflow flag may be incorrect
- ○ The output is the same only if the input values are positive

Which is not a benefit of floating point over fixed-point?

- ○ Floating point can represent smaller numbers
- ○ Floating point does not require more bits
- ○ Floating point can represent larger numbers
- ○ Floating point addition is simpler

Maximum marks: 6

14 Digital Logic



In the circuit above, 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?

- ◯ It will always produce the wrong results
- ◯ It will always produce the right results
- ◯ It will only produce the right results if A=1
- ◯ It will only produce the right results if A=0

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?

- ◯ The DRAM will be more sensitive to cosmic rays that leave random electrons on the chip.
- ◯ 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.
- ◯ All of the above
- ◯ 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.

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

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

Maximum marks: 6