Suggested Solutions

Part I: Fundamentals

1. Module 1: C and Assembly Programming

   (a) Short answer: 0xaf08fff0. The memory location (word) at address 0xff0 is modified.

   Elaborated answer: Encode the instruction using the MIPS sheet. Note that the `sw` instruction is an I-type instruction. The immediate value is calculated using two’s complement. The store word instruction stores a value to the memory. No registers are updated. The address for storing is calculated as 0x1000 - 1610 = 0xff0.

   Max 4 points. Three points if the machine encoding is correct. Remove one point for each incorrect hexadecimal digit. 1 point for correct memory location.

   (b) Short answer: Value 101 is printed to the standard output.

   Advice: To try out the solution, please copy and paste the code into a .c textfile, compile, and execute it.

2. Module 2: I/O Systems

   (a) Short answer: The code is as follows:

   ```c
   volatile int* control = (volatile int*) 0xffa0;
   volatile int* period = (volatile int*) 0xffa2;
   *control = (*control & ~0xc0) | (2 << 6);
   *period = 250;
   ```

   Some comments: The 8-bit period register cannot have a counter value that is larger than 255. The processor runs at 1MHz, which means that we want the timer to count to 1000 to make the timer period 1ms. We can set the prescale factor to 1:4 and then set the period register to 250. As an alternative, we can use prescale factor 1:8 and then set the period register to 125. This gives the same result.

   Max 4 points. Two points if the control register is correctly set. Minus one point if there is a small mistake with the masking. Two points if the period register is set correctly and that it matches the prescale factor.
(b) Short answers:

i. False. GPIO pins can be configured to be either input or output.
ii. True.
iii. False. Interrupts can be used for triggering external events and software interrupts can be used for system calls.
iv. True.

Max 4 points. 1 point for each correct answer, including motivation.

3. Module 3: Logic Design (for IS1500 only)

(a) Short answer: The set of possible values are as follows

- Case $C = 0$: Values 1, 3, 5
- Case $C = 1$: Values 1, 2, 4, 5, 7

As an alternative, the question may also be interpreted as what the values are exactly at 1000 ticks. In such a case, you need to look at the series of values. Then for case $C = 0$, the value is 3, and in case $C = 1$, the value is 5. We treat both interpretations as valid solutions.

Advice: If you cannot figure out the solution now after the exam, try to model the circuit in Logisim and check the result.

Max 4 points. 2 points for each correct case. One point for each case, if all but one numbers are correct.

(b) Short answer: A NAND gate with 3 inputs.

Max 2 points. Two points for the correct answer.

(c) Short answer: It is a Mealy machine. You should remove wire $B$ to get a Moore machine.

Max 2 points. One point for stating that it is a Mealy machine. One point for stating that it is wire $B$.

4. Module 4: Processor Design

(a) Short answer: $A = 0x7$, $B = 0x0$, $C = 0x0$, $D = 0x04010010$, and $E = 0x4$.

Elaborated answer:

Signal $A$ is the output from the register file from the second port. The address given to this port is value 8, which corresponds to register $t0$. By considering the value of $t1$ we see that this is the third time that we reach instruction bne. Hence, $A = 0x7$.

Signal $B$ is the register number for the first operand. In this case, the register is $zero$, so $B = 0x0$.

Signal $C$ states if the relative branch should be taken or not. Since $t0$ is not zero, we have $C = 0x0$.

Signal $D$ states the address to which the instruction should go if the branch is not taken. This is the instruction that comes after bne which is located 16 bytes after sw. Hence, $D = 0x04010010$.
Signal $E$ is the immediate value, shifted two steps to the left. By using equation $BTA = PC + 4 + \text{signext}(\text{imm}) \times 4$, we see that $E$ is part of the equation, $BTA = PC + 4 + E$. Hence, we have $E = BTA - PC - 4$. The current program pointer is $0x0401000C$ and the BTA is $0x04010014$. Hence, $E = 0x4$.

Max 5 points. One point for each correct number.

(b) Short answer:

\[
\begin{align*}
\text{l} & \quad \text{lw} \quad \text{t}0, 0(\text{t}1) \\
\text{a} & \quad \text{addi} \quad \text{t}2, \text{t}0, 1
\end{align*}
\]

Elaborated answer: In this case, it must be a load word (or load byte instruction) that writes the result to $\text{t}0$. The base register for specifying the memory address can be any register.

Max 3 points. Give 2 points if there is a clear dependency to $\text{t}0$. Give 3 points if it is completely correct, using the load word or the load byte instruction.

5. Module 5: Memory Hierarchy

(a) Short answer: The cache block size is 16 bytes, the tag field size is 21 bits, and there are 512 valid bits.

Elaborated answer: Each way holds $8192/4 = 2048$ bytes. Since the set field is 7 bits, there are 128 sets. Hence, the block size is $2048/128 = 16$ bytes. Since the block size is 16 bytes, 4 bits are needed for handling the block and the byte offset. As a consequence, the tag field is $32 - 7 - 4 = 21$ bits. Finally, each block needs a valid bit of its own. Hence, the number of valid bits are $8192/16 = 512$.

Max 3 points. One point for each correct answer.

(b) Short answer:

Line 1: \text{lui $t1, 0x1001} \quad \text{Instruction cache miss, no memory access}

Line 2: \text{lw $t0, 0x8(}$\text{t}1\text{)} \quad \text{Instruction cache hit, data cache miss}

Line 3: \text{lw $t0, 0x4(}$\text{t}1\text{)} \quad \text{Instruction cache hit, data cache hit}

Line 4: \text{lw $t0, 0x20(}$\text{t}1\text{)} \quad \text{Instruction cache miss, data cache miss}

Line 5: \text{lw $t0, 0x4(}$\text{t}1\text{)} \quad \text{Instruction cache hit, data cache hit}

Max 5 points. One point for each correct line.

6. Module 6: Parallel Processors and Programs

(a) Short answer: 2.5 seconds.

Elaborated answer:

Using Amdahl’s law, you realize that the part that can be parallelized approaches zero when the number of cores approaches infinity. Hence, we have $5 = \frac{10}{T_{\text{unaffected}}}$. This means that the part that is unaffected always takes 2 seconds. Hence, we can directly compute the speedup when using 4 cores: $\frac{10}{\frac{4}{2} + 2} = \frac{10}{4} = 2.5$.

Max 3 points. Three points if the answer is correct, else zero points.
(b) Short answer:

i. False. Moore’s law still holds (but we do not know for how long). However, it does not concern the power problem, it state that the number of transistors double every 18-24 months.

ii. True.

iii. False. SIMD instructions can also greatly affect performance.

iv. True.

v. True.

Max 5 points. One point for each correct answer, including motivation.
Part II: Advanced

7. A possible solution is as follows:

```c
void memmove(char* dest, const char* src, unsigned int size){
    /* Safe overlapped copying if the destination address
       is greater than the source. */
    if(dest > src && dest-src < size){
        for(int i=size-1; i >= 0; i--){
            dest[i] = src[i];
        }
    }
    /* Safe copying for non-overlapping memory regions */
    else{
        int i=0;
        /* Copy the header bytes (one byte each time)
           (max 3 bytes, until we are word aligned to destination) */
        for(;(i+(unsigned)dest)%4 != 0 && i<size; i++)
            dest[i] = src[i];

        /* Copy word by word, if both the source and the
           destination are word aligned */
        if((i+(unsigned)dest)%4 == 0 && (i+(unsigned)src)%4 == 0){
            int len = ((size - i)/4)*4;
            for(; i<len; i += 4)
                *(unsigned*) (dest+i)) = *((unsigned*) (src + i));
        }
        /* Copy the tail (byte by byte) */
        for(; i<size; i++)
            dest[i] = src[i];
    }
}
```

Max 20 points. Up to 10 points if the solution is correct for non-overlapping memory regions. Up to 15 points if the solution is completely correct, including overlapping memory regions. Up to 20 points if the solution is both completely correct and optimized in the case of non-overlapped memory regions.
Hi!

Thanks for your email. I also have some problems with the exam, but I think I know the answer to your questions. Please let me know if the following answers your questions.

- A stack is a data structure where data items can be pushed and popped. The things that are pushed (added to the stack) will first be popped (removed from the stack) last. When a function call is performed, registers must be push because the called function may change these registers. Other “things” that can be stored on the stack are, for instance, argument values and local variables.

- Machine code and assembly instructions are not the same thing, although very close. An assembler (a special program) can take assembly instructions (text) and translate that into machine code (binary data). Assembly code is text code that humans can understand, whereas machine code is binary data that the processor can decode. However, it is often very simple to translate from an assembly instruction to the corresponding machine code.

- You have unfortunately misunderstood the concept of IPC. IPC does not measure the latency of the pipeline, but rather how many instructions (on average) that can be performed every clock cycle. This is a very important component when evaluating the performance of a processor. However, it is not the only important factor. The clock frequency is equally important, so you cannot say that IPC or clock frequency is more important than the other. Clock frequency is very easy to advertise (it is just a number), whereas IPC is harder to grasp and depends on the benchmark that is executed.

- The main reason is due to the power wall that we hit around year 2006, that is, we cannot constantly increase the clock frequency and get higher performance: the processor simply gets too hot. Instead, processor manufactures decided to increase performance by putting several processors on the same chip, which is called multicore. Because of Moore’s law, we can put more and more transistors on a chip, which is why we can have multicore systems today that was not possible 20 years ago. The concept of using several processors was known already in the 90’s, but manufactures did not have enough resources on the chip and the processor got faster anyway, just by increasing the clock frequency.

- Unfortunately, I think you have misunderstood the concept of caches significantly. A cache is used by the processor to get faster access to memory. The reason is locality, both temporal and spatial locality. That is, if the same address is accessed often or if addresses close to each other are accessed. The problem with cache coherence is a special problem that occurs in multicore systems with shared memory and separate caches for the different cores. The coherence problem can result in that different cores see different values for the same address.

I hope that the above can be to some help. Good luck on the exam!

/David

Max 15 points. Max 3 points for each good answer.
9. For this exercise, we do not provide any actual concrete solution because the design choices and the explanations can very significantly. Please see the lecture slides and the course book for more information about the different concepts.

Max 20 points. At most 5 points for each section. To get full point for a section, all concepts must be clearly explained. Clear design choices must be given related to the concepts and it must be clear why these choices are preferable over the other alternatives.