# Written Exam / Tentamen

Computer Organization and Components / Datorteknik och komponenter (IS1500), 9 hp
Computer Hardware Engineering / Datorteknik, grundkurs (IS1200), 7.5 hp

**KTH Royal Institute of Technology**

2020-03-11

08.00-13.00

## Suggested Solutions

# Part I: Fundamentals

In part I, on the real exam, only short answers are expected. The elaborated answers given here are just for your information, and are not needed on the real exam.

1. **Module 1: C and Assembly Programming**

   (a) Short answer:

   ```
   lw    $s3,-8($v1)
   ```

   Max 3 points. One point for stating `lw`, one point for the correct immediate value ($-8$), and one point if both registers are correct.

   (b) Short answer: Function `foo` can, for instance, be defined as follows:

   ```
   void foo(int* p, int v){
     while(*p != 0){
       *p = *p + v;
       p++;
     }
   }
   ```

   Max 5 points. Give max 2 points for the correct signature. Remove one point if the incorrect return type is given or if any of the parameters are incorrect. Give max 3 points for the body, including the loop. Both a `while` or a `for` loop is OK. Remove one point for each part that is incorrect in the loop, e.g., missing to increment the pointer, incorrect dereferencing etc. If array indexing is used inside the body of the function, give 0 out of 3 points for the body.

2. **Module 2: I/O Systems**

   (a) Short answer: $50\,000$

   Elaborated answer: Taking the prescaling factor into account, the timer is incremented $80\,000\,000/8 = 10\,000\,000$ times every second. If we divide by 200, we know how many times the timer needs to be incremented within 5ms: $10\,000\,000/200 = 50\,000$. Hence, the period register should be configured to value $50\,000$.

   Max 2 points. Two points for a completely correct answer.

(b) Short answer:

```
    lui   $t0,0x1788      # setup address to LED lights
    ori   $t0,$t0,0x0010
    addi  $t3,$0,0
loop:
    lw    $t1,0xe440($0)  # get the status value of the buttons
    andi  $t2,$t1,0x80    # clear bits, execpt bit for button A
    beq   $t2,$0,notA
    addi  $t3,$0,13       # value 13, because button A was pushed
notA:
    andi  $t2,$t1,0x10    # clear all, execpt the bit for button D
    beq   $t2,$0,notD
    addi  $t3,$0,0        # save value 0, i.e., turn off all LEDS
notD:
    sll   $t3,$t3,5       # shift left because index 12 to 5
    sw    $t3,0($t0)      # write to LEDs
    j     loop            # repeat loop forever
```

Max 6 points. One point for using the correct address to the LED lights. One point for a correct infinite loop. One point if the program correctly retrieves the status of the buttons. One point if the program decodes button A correctly, and saves value 13. One point if it decodes button D correctly, and clears the bits. One point for shifting the LED value five bits to the left, and for using store word correctly. Remove one point if one requirement is not fulfilled, e.g. when pressing A nd D at the same time. Note that it is allowed to use pseudo instructions in the solution.

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

(a) Short answer: The sum-of-products form is $\overline{A}\,\overline{BC} + \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C}$

Max 2 points. Two points if correct (can be in a different form). One point if there is at most one error in one of the products.

(b) Short answer: 64 bits.

Elaborated answer: Since the address is 4 bits, we know that we have $2^4 = 16$ number of registers. Hence, each register can store $128/16 = 8$ bytes. As a consequence, the write port must be 64-bit wide (8 bytes).

Max 2 points. Two points for a correct answer.

(c) Short answer:

| *clock* | *A* | *B* | *Q* | *D* | *Y* |
|---|---|---|---|---|---|
| init | 1 | 1 | 1 | 1 | 0 |
| 1st | 0 | 0 | 1 | 0 | 1 |

Max 4 points. One minus point for each incorrect or missing number.

4. **Module 4: Processor Design**

   (a) Short answer: There is only one data hazard:

      i. Between `addi` and `sw`

      ii. Register `$s1`

      iii. Forwarding

      Max 3 points. One point for each correct answer. If more than one data hazard is given, reduce one point for each incorrect hazard.

   (b) Short answer: $A =$ `0x12`, $B =$ `unknown`, $C =$ `0x1`, $D =$ `0x220011`, and $E =$ `0x40004018`.

      Max 5 points. One point for each correct answer. Only hexadecimal values and the answer `unknown` are accepted as answers.

5. **Module 5: Memory Hierarchy**

   (a) Short answer:

      i. 2 instructions

      ii. 64 sets

      iii. 23 bits

      iv. 256 valid bits

      Elaborated answer: The block size is $2048/256 = 8$ bytes. Hence, two instructions are fetched at the same time if there is a cache miss. There are $256/4 = 64$ sets in the cache. Hence, the set field size is $6$ bits ($2^6 = 64$). The block size field is $3$ bits ($2^3 = 8$). Hence, the tag is $32 - 6 - 3 = 23$ bits. There is always one valid bit per block. Hence $256$ valid bits.

      Max 4 points. One point for each correct answer.

   (b) Short answer: (i) $\frac{1}{3}$ and (ii) 100%

      Elaborated answer: The block size of the instruction cache is $4096/256 = 16$ bytes. Hence, $4$ instructions are fetched when we have a cache miss. When the `addi` instruction is fetched at address `0x40001004` we get a cache miss. However, this block contains both he instruction before `addi` and the two `lw` instructions. We have 1 cache miss, and three memory accesses. Hence, the instruction cache miss rate is $\frac{1}{3}$.

      The block size of the data cache is $4096/512 = 8$ bytes. We have two data accesses using the two `lw` instructions. Hence, we have 2 data accesses. The first data access gives a cache miss when it accesses address `0xe54`. However, the block that is fetched does not include address `0xe58` which is the second memory access. Hence, we get two misses, and a miss rate of $\frac{2}{2} = 1$, i.e., 100%.

      Max 4 points. 2 points for each miss rate.

6. **Module 6: Parallel Processors and Programs**

   (a) Short answers:

      i. False. In data-level parallelism, each instruction operates on several data items.

      ii. False. AVX is standard for SIMD instructions and has nothing to do with hardware multithreading.

      iii. True.

      iv. True.

      v. False. False sharing means that two different threads use the same cache line, which results in invalidation of the cache. It has nothing to do with virtual memories.

      Max 5 points. One point for each correct answer. The false statements need to have clear motivations.

   (b) Short answer: 80%

      Elaborated answer. Using Amdahl's law, we know that the theoretical speedup is achieved when we have infinitely many cores, that is, when the parallelizable part's execution time goes to zero. Hence, the sequential part is $\frac{30}{5} = 6$s. This means that the sequential part is $\frac{6}{30} = 20\%$. Hence, the parallel part is $80\%$.

# Part II: Advanced

7. The complete solution is omitted. Please see the lecture slides and the course book.

   The question is graded in three levels S, G, and VG, with the following criteria:

   - Satisfactory (S): Some of the concepts in the question are clearly explained.
   - Good (G): Basically all concepts in the question are clearly explained, and some of the concepts are related to each other, by discussing similarities and differences.
   - Very Good (VG): Basically all concepts in the question are clearly explained, and all of the concepts are related to each other, by discussing similarities and differences.

   If none of the three levels is achieved, the exercise is considered as failed (F).

8. The following C code shows an example of a C code solution.

```
void myprint(int xpos, int ypos, const char* text){
  int char_no = 0;
  while(*text != 0){
    char* spos = screen + xpos + ypos * S_WIDTH + F_WIDTH * char_no;
    int charval = (int) *text - 65;
    char* fontp = font + charval * F_WIDTH;
    for(int i=0; i<F_HEIGHT; i++){
      for(int j=0; j<F_WIDTH; j++){
        *spos++ = *fontp++;
      }
      spos += S_WIDTH - F_WIDTH;
      fontp += F_WIDTH * (F_CHARS - 1);
    }
    char_no++;
    text++;
  }
}
```

   The question is graded in three levels S, G, and VG, with the following criteria:

   - Satisfactory (S): Some minor parts of a C or Assembly program are constructed correctly, but there are major errors or missing parts of the program.
   - Good (G): The majority of the program is constructed correctly, including the main flow and structure of the program, but there are a number of minor errors within the program.
   - Very Good (VG): The program is correct, with basically no errors.

   If none of the three levels is achieved, the exercise is considered as failed (F).

9. Potential solutions for the three different sub-problems L1, L2, and L3 are given below.

   - L1: By inspecting the program, we see that we have two elements 10 and 7 in the array. The `sw` instruction is executed twice, when swapping the two elements.
   - L2: All data hazards can be solved by using forwarding, execpt in the case of one of the two `lw` instructions. The values that are loaded using `lw` are stored in registers `$t5` and `$t6`. They are then used in the instruction `slt $t3,$t5,$t6`. By inspecting the pipeline stage, we can see that the first `lw` has enough spacing, such that it can still be solved by only using forwarding. However, between the second `lw` and the `slt` instruction, a stalling is needed. The involved register is `$t6`.
   - L3: The block size is $4096/256 = 16$ bytes. Hence, each cache block handles 4 instructions. The first `addi` instruction in `sort` is the last instruction in its cache block. However, when the `main` function executes before calling `sort`, this cache block has already been fetched. Hence, the first `addi` instruction in the `sort` function results in a cache hit. The next `addi` instruction results in a cache miss, and it fetches 4 instructions. The next miss is the `slt` instruction. By inspecting the whole flow, we can see that all instructions in the function are executed at least once. Hence, it is enough to count the number of cache blocks that are fetched because of cache misses. By inspecting the whole program, we can see that we have 5 misses in total.

   The question is divided into three analysis tasks, each corresponding to one of three levels of difficulty: L1, L2, and L3. The question is graded in three levels S, G, and VG, with the following criteria:

   - Satisfactory (S): The task at level L1 is solved correctly.
   - Good (G): Either the task at level L2 or the task at level L3 is solved correctly.
   - Very Good (VG): Both the tasks at level L2 and L3 are solved correctly.

   If none of the three levels is achieved, the exercise is considered as failed (F).

## Correction of the Exam

The examiner David Broman authored the exam questions, the correction guidelines, and the suggested solutions. The following persons took part in the correction: Saranya Natarajan, Viktor Palmkvist, Daniel Lundén, Linnea Ingmar, and Fredrik Lundevall.