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

2019-03-15

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: `0x00094083` and -8.

   Note: `sra` shifts n bits to the right. If the most significant bit is 1, ones are inserted to the left, and if the most significant bit is 0, zero values are inserted. Hence, it performs a signed division, by dividing the original value with 2^n. In this case, dividing by $2^2 = 4$.

   Max 4 points. Three points for completely correct machine code, two points if at most one byte is incorrect, and one point if at most two bytes are incorrect. One point for the correct decimal number.

   (b) Short answer:

   The two lines of C code are

   ```
   *p = *p + 2;
   p++;
   ```

   and the array content is as follows.

   ```
   {9, 10, 12, 0}
   ```

   Max 4 points. One point for dereferencing correctly and writing using `*p`. One point for dereferencing and reading and adding correctly, using `*p + 2`. One point for incrementing the pointer correctly. One point for writing out the correct array content (if the last zero value is not given, we still give one point).

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

   (a) Short answer:

   Elaborated answer:

   i. True.

   ii. False. The `volatile` keyword in C can be used to declare a volatile pointer, which means that the content to which the pointer points to can be changed at any time. It has nothing to do with volatile memories.

   iii. True.

   iv. False. An interrupt makes it possible to interrupt and execute some other piece of code, and then safely go back to the original code and continue execution.

   Max 4 points. One point for each correct answer. The false statements need to have clear motivations.

   (b) Short answer:

```
lui   $t0,0x9500       # Get the correct address into $t0
ori   $t0,$t0,0x3f00

lw    $t1,0($t0)       # Load the current value of the port

lui   $t2,0xffff       # Mask the 7 LED bits
ori   $t2,$t2,0xfc07
and   $t1,$t1,$t2

ori   $t1,$t1,0x278    # Or on the bits that represent value 3

sw    $t1,0($t0)       # Store the new value
```

   Max 4 points. One point for correctly setting the port address. One point for loading and correctly storing back the value. One point for correctly masking the value (using `and`). One point for correctly using an OR instruction to add the correct bits. If the correct bits are written to the port, but the bits that are not used by the LEDs are destroyed, max 2 points can be given in total.

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

   (a) Short answer:

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

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

(b) Short answer:

A	B	C	$C \cdot \overline{A \cdot B}$	$C \cdot \overline{A}$	$C \cdot \overline{B}$	$C \cdot \overline{A} + C \cdot \overline{B}$
0	0	0	0	0	0	0
0	0	1	1	1	1	1
0	1	0	0	0	0	0
0	1	1	1	1	0	1
1	0	0	0	0	0	0
1	0	1	1	0	1	1
1	1	0	0	0	0	0
1	1	1	0	0	0	0

Comment: For all cases, we show in a truth table that the values on the left hand side (column 4) and right hand side (column 7) are equal. A complete solution do not have to include the extra columns (5 and 6).

Max 3 points. One point if there is a truth table with 8 cases, including A, B, and C. Two points if the truth table is almost correct, with at most 4 errors. 3 points if the truth table is completely correct.

(c) Short answer: Alternative i)

Max 1 points. One point if the correct alternative.

4. **Module 4: Processor Design**

(a) Short answer: $A =$ `0x4`, $B =$ `0x2f`, $C =$ `0x34f8`, $D =$ `unknown`, and $E =$ `0x0`.

Max 5 points. One point for each correct value.

(b) There are two hazards. The first hazard is between the first `addi` instruction and the `sll` instruction and involves register `$t0`. This hazard can be solved using forwarding. The second hazard is between the `sll` instruction and the `lw` instruction. It involves register `$t2`. It can also be solved using forwarding.

Max 3 points. One point for each of the two hazards (stating both between which instructions and stating the register involved). One point for stating that both can be solved using forwarding. At most two points are given if statements are given about incorrect hazards.

5. **Module 5: Memory Hierarchy**

(a) Short answer: i) 128 sets, ii) 1024 bytes, and iii) `0x94f8`.

Elaborated answer:

i. The byte offset field size is 3 bits ($2^3 = 8$) and the tag size is 6 bits. Hence, the set field is $16 - 6 - 3 = 7$ bits. Hence, there are $2^7 = 128$ sets.

ii. The capacity is $128 \cdot 8 = 1024$ bytes.

iii. The address must have the binary tag `100101`. The set must have value 31, which in binary form is `0011111` (7 bits for the set field). Finally, any value for the byte offset is OK for a cache hit. Let us select zero values for the 3-bit byte offset field. Hence, an address is in binary form `1001010011111000`, which is the same as `0x94f8`.

Max 4 points. One points for each of the two first questions. Two points if the address is correct, and 1 point if there is at most one digit incorrect in the address. Note that there are many possible values for the byte offset field.

(b) Short answer:

  i. 1: miss, 2: miss, 3: hit, 4: hit, 5: hit, 6: miss, 7: hit.
  ii. The hit rate is $\frac{4}{7}$.

Note that it does not matter if the cache is direct mapped or if it is an N-way set associative cache where $N > 0$. The results in this exercise are the same.

Max 4 points. The first question gives 3 points if everything is correct. It gives 2 points if at most 2 values are incorrect. It gives 1 point if at most 4 values are incorrect. The second question gives 1 point if it is completely correct.

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

(a) Short answer: $\frac{5}{2}$

Elaborated answer: Using Amdahl's law, we have

$$5 = \frac{10}{\frac{10-x}{N} + x}. \tag{1}$$

If $N \to \infty$, then $x \to 2$. Hence, the sequential part of the program is 2s. This means that the part that can be parallelized takes 8s on one core. Hence, we can calculate the speedup where $N = 4$ by using Amdahl's law again.

$$\frac{10}{\frac{8}{4} + 2} = \frac{10}{4} = \frac{5}{2}. \tag{2}$$

Max 3 points. Three points for the correct value.

(b) Short answers:

  i. True.
  ii. False. A *mutex* is useful for this process, not a *multiplexer* (which is used in digital design).
  iii. True.
  iv. True.
  v. True. (Note that this question is also OK to answer false, as long as the argumentation is correct. That is, the argument can either be that it needs to be *hardware* multithreading, or that it does not have to be *fine-grained* multithreading.)

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

Part II: Advanced

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

   Max 15 points. For each of the three items, max three points for the explanations of the two concepts, max one point for at least one difference, and max one point for at least one similarity. That is, max five points for each of the three items.

8. In the following solution, we assume that the addresses for `foo`, `lst`, `s1`, and to the stack pointer `$sp` all reference the first element in a cache block. We also assume the strategy predict branch not taken. A solution with other assumptions can also be valid, as long as the assumptions are clearly stated.

   (a) Each cache block can load 4 instructions and the first instruction of `foo` starts at an even address. Note that the first `bne` instruction results in a jump to the `else` label. The `while` loop will loop once. The function will call itself once, but then return directly using the first `ra` instruction. Counting the number of misses each time a new block is accessed, we get the following number of misses: $1 + 1 + 1 + 1 + 1 + 0 + 0 + 0 + 1 = 6$ misses. The number of instruction fetches are $4+1+2+4+1+1+2+4+4+2+2+3 = 30$. Hence, the cache miss rate is $\frac{6}{30} = \frac{1}{5}$.

   (b) The first time the stack is accessed when coming into `foo`, we get one miss out of two memory accesses. The first `lw` results in a miss. In the while loop, there will be two accesses using `lb`, where the first one results in a miss, and the second one a hit. The next time `foo` is called recursively, the two `sw` result again in first a miss and then a hit. The following three `lw` all result in hits. After coming back to the previous call to `foo`, the final two `lw` result in hits. In summary, we have $1+1+1+1+0+0 = 4$ misses, and $2+1+2+2+3+2 = 12$ memory accesses. Hence, the data cache miss rate is $\frac{4}{12} = \frac{1}{3}$.

   (c) We already know that the number of clock cycles are equal to the number of fetches, plus penalties. The number of fetches are (from part 8a) 30, i.e. 30 clock cycles.

   For data hazards, we only have to care about hazards that need to be solved using stalling. This happens two times between `lw` and the `bne` instruction (in the beginning of `foo`, each time `foo` is called). It also happens two times between the `lb` and the `beq` instruction. Hence, we have 4 extra penalty clock cycles for stalling due to data hazards.

   For control hazards, we get a 3 clock cycle penalty if a branch is taken (the check is done in the execute stage). The first `bne` results in 3 clock cycle penalty. The `j` instruction results in one penalty, and the following `beq` to label `exit` results in 3 more clock cycles. The `jal` results in one penalty, and the `jr` results in one penalty. Hence, in total, we have $3+1+3+1+1 = 9$ clock cycles penalty because of control hazards.

   Finally, we know that we have 6 instruction cache misses and 4 data cache misses. This means that we get $(6 + 4) \cdot 10 = 100$ clock cycles because of cache misses.

   In summary, we have $30 + 4 + 9 + 100 = 143$ clock cycles in total.

   Max 18 points in total.

9. (a) A possible C code implementation is as follows.

```
int foo(char **p){
  if(*p == 0){
    return 0;
  }
  else{
    char *sp = *p;
    int len=0;
    while(*sp){
      len++;
      sp++;
    }
    return foo(p+1) + len;
  }
}
```

(b) The program prints out `Length 9`. The program computes the total number of characters (excluding the 0 termination values) of all strings pointed to by parameter `p`.

Max 17 points in total.

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, Elias Castegren, Daniel Lundén, and Fredrik Lundevall.