KTH Information and
Communication Technology

IL2206 EMBEDDED SYSTEMS

Exercise Collection

August 17, 2015

Contents

1 Programming

1-1: Write the Nios II assembly code needed for the following faculty function. Follow the *Nios II Application Binary Interface* (ABI) so that the function can be called from a C program. Document your code.

```
int fac(int x)
{
  int f;

  if (x <= 0)
    f = 0;
  else {
    f = 1;
    while(x > 1)
    {
      f = f * x;
```

```
      x = x - 1;
    }
  }
  return f;
}
```

1-2: Given is the following program:

```
#include <stdio.h>

int main(void)
{
  char *cp;
  short *sp;
  int *ip;
  short x[6];

  int i, y;

  y = 0x0102;
  for (i = 0; i < 6; i++) {
      x[i] = y;
      y = y + 0x1010;
  }
  cp = (char*) x;
  printf("1) *cp   = %x\n", *cp);
  sp = x;
  printf("2) *sp   = %x\n", *sp);
  printf("3) cp[3] = %x\n", cp[3]);
  ip = (int*) x;
  ip = ip + 1;
  printf("A) *ip   = %x\n", *ip);
  printf("B) cp[6] = %x\n", cp[6]);
  sp = sp + 5;
  printf("C) *sp   = %x\n", *sp);
  *x = *cp + 2;
  printf("D) cp[1] = %x\n", cp[1]);
  return 0;
}
```

Execution of the code gives the following output:

```
1) *cp   = 2
2) *sp   = 102
3) cp[3] = 11
A) *ip   = ??
B) cp[6] = ??
C) *sp   = ??
D) cp[1] = ??
```

Fill in the output for A), B), C), D). Assume the following sizes for the variables, `int`: 4 bytes, `short`: 2 bytes, `char`: 1 byte.

1-3: The following code shall be executed in Ada 2005. Give all possible results for the variable `N`.

```
with Ada.Text_IO; use Ada.Text_IO;
procedure Concurrency is
   N : Integer := 1;
   task Task1; task Task2; task Task3;
   task body Task1 is
   begin
     N := N * 2;
   end Task1;
   task body Task2 is
   begin
      N := N + 3;
   end Task2;
   task body Task3 is
   begin
      N := 1;
   end Task3;
begin
   delay 1.0; -- Wait 1 second
   Put_Line("N = " & Integer'Image ( N ));
end Concurrency;
```

1-4: A client-server application shall be implemented in Ada 2005. The server shall monitor the number of requests from the clients. For this it uses an internal variable `N`. For each request `N` is incremented and the current number of requests is returned to the client.

(a) Which Ada interprocess communication mechanism would you use?

(b) Give the Ada code for the client.

(c) Give the Ada code for the server.

2 Embedded Computing Platform

2-1: A peripheral circuit shall be connected to a CPU with an eight-bit address and data bus by memory-mapped I/O. The peripheral circuit has eight registers, which are controlled by the inputs $\mathtt{RS}_2$ to $\mathtt{RS}_0$. The circuit has also a chip select input `CS`, a read/write input $\mathtt{R}/\overline{\mathtt{W}}$ and eight data bits $\mathtt{D}_7$ to $\mathtt{D}_0$.

Show with a diagram how the peripheral circuit can be connected with the CPU, so that the eight registers are accessed only via the adresses 0x10 to 0x17.

2-2: In order to fulfill the design requirements, a system needs an average memory access time of 8 ns, the cache access time is 5 ns and a main memory access time is 70 ns.

(a) Which cache hit rate is needed to achieve this performance?

(b) Which cache hit rate would be needed for an average access time of 15 ns?

(c) Which cache hit rate would be needed for an average access time of 6 ns?

2-3: A small *byte-addressable* embedded computer system, with a word length of 32 bits, has a main memory consisting of 4 KBytes. It also has a small data cache capable of holding eight 32-bit words, where each cache line contains only two words. When a given program is executed the processor reads data from the following sequence of hex addresses:

0x010, 0x1FC, 0x168, 0x008, 0x014, 0x1F8, 0x00C.

This pattern is repeated four times.

(a) Consider a direct-mapped cache.

i. How many bits are used for the tag, block, and offset fields for the representation of a memory address?

ii. Show the contents of the cache at the end of the execution. Assume the cache is empty at the start of the program.

iii. Compute the hit-rate for this example. Assume that the cache is initially empty.

(b) Consider a 2-way set associative cache with the same cache size (eight words of 32 bits) and block size (2 words in a block).

i. How many bits are used for the tag, set, and offset fields for the representation of a memory address?

ii. Give an example for a sequence of read accesses, where the 2-way set-associative cache memory performs much better than the direct-mapped cache.

2-4: An embedded processor system has three masters and four slaves. The processor system shall run the following abstract parallel program that only shows the order of slave accesses.

M_1	M_2	M_3
ReadS_1	ReadS_2	ReadS_4
ReadS_1	ReadS_3	ReadS_3
	ReadS_2	

To simplify we assume that a read transaction takes one cycle.

(a) Assume that master and slaves are connected by a classical bus, i.e. only one master can access the bus during a bus cycle. How many cylces are needed to execute the abstract program?

(b) Assume that the processor system uses weighted round-robin slave-side arbitration as implemented in the Avalon Switch Fabric. The weights are given below. A '-' means that the master and slave are not connected.

	S_1	S_2	S_3	S_4
M_1	1	-	-	-
M_2	-	1	2	-
M_3	-	-	1	1

i. Draw a schematic of the switch fabric described by the table.
ii. For all slaves, give the percentage of accesses that each master is guaranteed during a longer time period.
iii. Give the best case execution time for the abstract program.
iv. Give the worst case ecxecution time for the abstract program.

2-5: A multiprocessor system consisting of three processors and a shared memory uses slave-side arbitration with weighted round-robin for the memory access. The processors have the following weights: $wt_1 = 1, wt_2 = 2, wt_3 = 3$. A full read or write memory transaction is assumed to take exactly one memory cycle, which lasts 100ns. We also assume that an instruction that does not include a memory transfer is completed in 20ns.

For this task we assume that a master receives never more shares than the number given by its weight within one round. Further we assume that in each round the processors are served in order starting with processor 1.

The following code, which uses the Altera Nios II instruction set, shall be run on processor 1 and 3. Processor 2 is assumed to be idle and will not access the memory. The variables `x` and `y` are shared variables.

- Processor 1

```
       ...
a:
  movi  r6, 1         # x = 1;
  stwio r6, x
while:                # while (y != 1)
  ldwio r7, y         #     ;
  bne   r6, r7, while
b:
  movi  r8, 2         # z = 2;
  stw   r8, z
       ...
```

- Processor 3

```
       movi r6, 0           # y = 0;
       stwio r6, y
       ...
       movi r6, 1
while:
       ldwio r7, x          # while (x != 1);
       bne r6, r7, while    #     ;
       stwio r6, y          # y = 1;
       ...
```

Give the best case and worst case execution time for the program fragment for processor 1 that starts at label `a` and finishes before label `b`. At the time when processor 1 is at label `a`, processor 3 is at label `while`.

3 Real-Time Systems

3-1: Assume three periodic tasks $T_1 = (9,3)$, $T_2 = (2,1)$ and $T_3 = (6,1)$, which shall be scheduled on a single processor.

(a) Give the processor utilisation.

(b) Can the rate-monotonic algorithm produce a feasible schedule? In such a case, give a schedule.

(c) Can the earliest-deadline-first algorithm produce a feasible schedule? In such a case, give a schedule.

A task T is defined as (p,e), where e is the execution time and p the period. The relatvie deadline $d = p$!

3-2: Two tasks in a preemptive real-time operating system have both access to the shared variables `a` and `b`. Task 1 writes variable `a` and reads variable `b`. Task 2 writes variable `b` and reads variable `a`. Task 1 never writes variable `b` and task 2 never writes variable `a`. Given is the following code fragment. Task 1

```
...
a = f1(...);
// Synchronization Point
f2(b);
...
```

Task 2

```
...
b = g1(...);
// Synchronization Point
g2(a);
...
```

The tasks shall now be synchronized so that

- task 1 does not execute `f2(b)` before the statement `b = g1(...)` has been executed in task 2
- task 2 does not execute `g2(a)` before the statement `a = f1(...)` has been executed in task 1

Assume the operating system supports semaphores and provides the functions `accessSem(Sem)` and `releaseSem(Sem)` to access or release the semaphore `Sem`. Modify the code so that the tasks are synchronized as specified above. Draw also a diagram that illustrates how processes and semaphore(s) interact.

3-3: A task T is defined as (p,e), where p the period, e is the execution time, and the deadline $d = p$. Given are the following sets of tasks.

(1) $T_1 = (4,1)$, $T_2 = (5,2)$, $T_3 = (10,1)$.

(2) $T_1 = (3,1)$, $T_2 = (10,4)$, $T_3 = (20,6)$.

(3) $T_1 = (4, 1), T_2 = (2, 1), T_3 = (8, 1)$.

Determine for each set of tasks, if there exists a feasible schedule, if

(a) the tasks are scheduled according to the rate-monotonic algorithm.

(b) the tasks are scheduled according to the earliest deadline-first algorithm.

3-4: Given is the pseudo-code for the following three tasks T_1, T_2, and T_3. Task T_1 has the highest priority and task T_3 has the lowest priority. The tasks shall be executed using MicroC/OS-II, a preemptive real-time operating system, which uses non-cooperative multi-tasking.

```
void t1(...)
{
  while(1 < 2) {
    printf("1");
    OSSemPost(semA);
    OSSemPend(semB);
  }
}
void t3(...)
{
  while(1 < 2) {
    printf("3");
    OSSemPost(semB);
    OSSemPend(semA);
  }
}
```

```
void t2(...)
{
  while(1 < 2) {
    printf("2");
  }
}
```

Give the first five values that are output, if you execute the program. You can assume that all three processes are ready at program start and that both semaphores `semA` and `semB` are initialized with the value 0.

3-5: The following algorithm using a weak binary semaphore S that was initialized with $S = \{1, \emptyset\}$ solves the critical section problem for two processes.

P_1	P_2
`loop forever`	`loop forever`
`  non-critical section`	`  non-critical section`
`  wait(S)`	`  wait(S)`
`  critical section`	`  critical section`
`  signal(S)`	`  signal(S)`

(a) Show why it solves the problem for two processes.

(b) Does it also solve the critical section problem for three or more processes?

3-6: A set of n concurrent processes accesses a shared variable `Shared` of type integer. In order to provide consistency, mutual exclusive access to the variable is to be ensured during a read operation `ReadVariable` and a write operation `WriteVariable`.

(a) Give the pseudo-code for a protected object with the operations `ReadVariable()` and `WriteVariable(X: Integer)` that ensures mutually-exclusive access to the variable `Shared`.

(b) What property is required for the protected object, so that starvation does not occur?

3-7: How can the CPU utilization be measured in a priority-driven preemptive real-time operating system with fixed priorities such as MicroC/OS-II?

3-8: Given are the following two periodic tasks: $T_1 = (300, 100)$, $T_2 = (400, 200)$. The task T_1 executes periodically a single function f_1, which has an execution time that is always less than 100 time units. The task T_2 executes a single function f_2, which has an execution time that is always less than 200 time units.

Explain, how you can implement these periodic tasks using a hardware timer, and a preemptive real-time operating systems that provides soft timers and semaphores.

Illustrate your solution with a diagram containing hardware timer, soft timer(s), semaphores and tasks, and give pseudocode for both tasks T_1 and T_2.

3-9: Describe the principle operation of a *watch-dog timer*. Illustrate it with a block diagram.

4 Program Design and Analysis

4-1: Given are the following functions A and B.

```
\\Function A
int i;
int w[512];
...
for(i = 1; i < 25; i++) {
    w[159]++;
    w[160]++;
}
y = w[159] + w[160];
...
```

```
\\Function B
int i;
int w[512];
...
for(i = 1; i < 25; i++) {
    w[145]++;
    w[274]++;
}
y = w[145] + w[274];
...
```

Will one of the two functions A and B execute faster than the other function (assuming no optimization),

(a) if a direct-mapped cache with a size of 128 Bytes and a block size of 32 Bytes is used.

(b) if a direct-mapped cache with a size of 128 Bytes and a block size of 4 Bytes is used.

Assume that the variable `i` is stored in a register and that `w[0]` is stored in the memory location `0xA80` and that the size of an integer is 4 Bytes. Motivate!

4-2: Given is the following code.

```
u = c + d;
v = a - b;
w = a - u;
x = v + e;
```

(a) How many registers are needed for the following code, if no further optimisation is used?

(b) How can the code be improved so that a minimal number of registers is used?

Assume that the processor provides only the following type of operations $op\ R_1, R_2, R_3$, where $R_1 \neq R_2$ and $R_1 \neq R_3$ and $R_2 \neq R_3$.

4-3: (*Wolf: Computers as Components, Q5-16*) The loop appearing below is executed on a machine that has a 1K Word direct-mapped data cache with four words per cache line.

```
for (i = 0; i < 50; i++)
  for(j = 0; j < 4; j++)
    x[i][j] = a[i][j] * c[i];
```

Assume that all variables have the size of 4 Bytes.

(a) How must `x` and `a` be placed relative to each other in memory to produce a conflict miss every time the inner loop's body is executed?

(b) How must `x` and `a` be placed relative to each other in memory to produce a conflict miss one out of every four times the inner loop's body is executed?

(c) How must `x` and `a` be placed relative to each other in memory to produce no conflict misses?

4-4: Given is the following function. Assume that

- after line 4, the arrays `u` and `w` are located in sequence in the memory, first all elements of `u` and then all elements of `w`
- `u[0]` is mapped to the first word of the first cache line
- one int value consists of 32 bits
- in every execution of the loop, `u` is accessed before `w`

```
int i;
int y = 0;
int u[60];
int w[60];
...
for(i = 0; i < 60; i++) {
    y += u[i] * w[i];
}
...
```

Assuming that the variables `i` and `y` are stored in registers and no further optimization,

(a) Calculate the hit rate in the cache, if a direct-mapped cache with a size of 128 Bytes and a block size of 32 Bytes is used. Motivate!

(b) What is needed to maximize the cache hit rate and how can this be implemented? Which cache rate can be achieved? Is it possible to reach 100%? Motivate!

4-5: Given is the following code fragment.

```
#define MAX 10

int a[MAX], b[MAX], c[MAX], x[MAX], y[MAX];
int i, j, r, s;
...
int f(int a, int b) {
  int z;
  z = 2 * a - b;
  return z;
}
int g(int a, int b, int c) {
  int z;
  z = a * c - c* b;
  return z;
}
...
for(i = 0; i <= MAX - 1; i++){
  x[i] = f(a[i], b[i]);
}
s = 2 * r;
for(j = 0; j <= MAX - 1; j++){
  y[j] = s * g(a[j], b[j], c[j]);
}
...
```

Try to find optimizations that result in a shorter execution time, while not increasing code size. Explain each optimization and give the optimized code as C-code.

4-6: Given is the following code fragment.

```
y = 0;
if (mode == 1) {
   for (i = 0; i < 5; i++) {
      y += a[i] * b[i];
   }
}
```

(a) Draw the control and data flow graph (CDFG) for this code fragment.

(b) Determine the best case and worst case execution time, if the multiplication instruction takes three time unit, an addition takes one time unit and all other statements and branches take one time unit. The processor can only execute one instruction at each time instance. The processor does not have a cache.

(c) How much is the execution time reduced, if the processor has a special instruction *Multiply-Accumulate* that calculates `R1 = R1 + R2 * R3` in a single cycle (1 time unit)?

5 Fixed-Point and Floating-Point

5-1: Given is the following floating-point format $b_5b_4b_3b_2b_1b_0$, where the sign-bit is given by b_5, the mantissa by $1.b_2b_1b_0$, and the exponent by b_4b_3. The value is calculated as $-1^{b_5} * 1.b_2b_1b_0 * 2^{b_4b_3-1}$.

(a) Is it possible to represent the numbers 0.25, 0.8125, -1.375, 4.25 and 7.5 in this format? Motivate!

(b) Add the following numbers: $b_5b_4b_3b_2b_1b_0 = 001111$ and $b_5b_4b_3b_2b_1b_0 = 010010$. What would be needed to avoid loss of information?

(c) Multiply the numbers of the previous subtask.

5-2: Given are the following $Q3$-Numbers: $A = (0.110)_2$ and $B = (1.100)_2$.

(a) Add the numbers A and B

(b) Multiply the numbers A and B

In both cases the result shall be stored in $Q3$-format.

5-3: If 6 bits are available for a floating-point number, and one is used as sign-bit, is the dynamic range larger, if

(a) 3 bits are used for the mantissa and 2 bits for the exponent

(b) 2 bits are used for the mantissa and 3 bits for the exponent

Motivate!

5-4: Is it possible to represent the number $(0.7)_{10}$ without loss of precision in

(a) Q_n fixed-point format

(b) a floating-point representation with n bits $b_{n-1}b_{n-2}\dots b_1b_0$, where the value is calculated by $-1^{b_{n-1}} * (1.b_{k-1}\dots b_0)_2 * 2^{b_{n-2}\dots b_k-1}$.

Motivate your answer. If there is a representation in a format, give the values for a minimal n and k and the number in that representation.

5-5: Compare the following two formats

- Q3-fixed-point format

- floating-point format $b_3b_2b_1b_0$, where the sign is given by -1^{b_3}, the mantissa by $1.b_0$, and the exponent by b_2b_1. The value is calculated as $-1^{b_3} * 1.b_0 * 2^{b_2b_1-1}$.

For each of the representations

(a) give the numbers that can be represented with full precision.

(b) give the maximum quantisation error

(c) check, if the number 0 can be represented with full precision, and if not, suggest how this format can be changed so that 0 is represented.

6 Hardware Accelerators

6-1: A DSP application implemented on a uniprocessor uses a multiply accumulate subroutine 15% of the total execution time. What is the speed-up, if a new version of the processor would include a multiply accumulate instruction that only needs 10% of the execution time of the subroutine?

6-2: A system consists of a processor and an accelerator. They are connected by a shared bus with a memory. Each of them has many registers to store values. Communication between them is done via the shared memory.

Each `Load` and `Store` takes 2 time units. Register accesses inside the processor or accelerator do not consume time. An add operation can only be performed on the processor and takes 1 time unit. A multiplication takes 8 time units on the processor, but only 2 time units on the accelerator.

The following algorithm shall be implemented:

```
X = A * C + B * C + D
Y = (A + B + C) * D
```

At the start of the program the variables `A`, `B`, `C` and `D` are stored in memory and the results `X` and `Y` shall bestored in memory at the end of the algorithm.

Both processor and accelerator have `LOAD` and `STORE` instructions and transfer a memory value to a register and vice versa. The arithmetic instructions `ADD` and `MUL` take two register operands and store the result in a register. The instruction `ADD` is not available on the accelerator.

(a) How much time will the algorithm take, if is executed on the processor?

(b) How much time will the same algorithm take, if it is executed on processor and accelerator? How large is the speed-up?

In both cases show the schedule of the program.

6-3: A system consists of a processor and an accelerator. They are connected by a shared bus with a memory. Each of them has many registers to store values. Communication between them is done via the shared memory.

Each `Load` and `Store` takes 2 time units. Register accesses inside the processor or accelerator do not consume time. An add operation can only be performed on the processor and takes 1 time unit. A multiplication takes 8 time units on the processor, but only 1 time unit on the accelerator.

The following algorithm shall be implemented:

```
X = A * B + C * D
```

At the start of the program the variables `A`, `B`, `C` and `D` are stored in memory and the result `X` shall be stored in memory at the end of the algorithm.

Both processor and accelerator have `LOAD` and `STORE` instructions to transfer a memory value to a register and vice versa. The arithmetic instructions `ADD` and `MUL` take two register operands and store the result in a register. The instruction `ADD` is not available on the accelerator.

(a) How much time will the algorithm take, if is executed on the processor?

(b) How fast can the same algorithm be executed, if it is executed on processor and accelerator? How large is the speed-up?

In both cases show the schedule of the program.

7 Solution to Selected Exercises

This section will be updated after each tutorial session.