Final Exam

Topics not examinable:
- Gates and Flip-flops
- Memory systems

Structure of Final Exam

Four Questions:
- Question 1: Basic Concepts (20 marks)
  - 10 PARTs.
- Question 2: Miscellaneous Questions (25 marks)
  - 7 PARTs.
- Question 3: AVR Assembly Programming (30 marks)
  - 3 PARTs.
- Question 4: Interrupts and I/O (25 marks)
  - 6 PARTs.
Sample PARTs in Question 1

Discuss the advantages of using macros versus functions. (2 marks)

Describe how the prioritisation is achieved in daisy chain interrupts. (2 marks)

Sample PARTs in Question 2

Consider the following program:

```
ldi r0, 1
clr r1
clr r2
clr r3
loop: cpi r0, 201
breq end
add r2, r0
adc r3, r1
inc r0
jmp loop
end: nop
```

If the program is executed on an AVR microcontroller with a clock frequency of 8 MHz, what is its execution time? (3 marks)
Consider two single precision floating point numbers \( x \) and \( y \) in IEEE 754 format, where \( x = 0xC0EE8000 \) and \( y = 0xC1AA4000 \). What is the hexadecimal value of the result of \( x + y \) in IEEE 754 format? (3 marks)

Consider the following AVR assembly code.

```
.set x=pc
ldi r20, low(x)
ldi r21, high(x)
ldi r30, low(pc)
ldi r31, high(pc)
```

Assume that the address of the instruction “ldi r20, low(x)” is 0x0202. After the above sequence of code is executed, what are the contents in r20, r21, r30, and r31, respectively? (3 marks)

A C program consists of five functions. Their calling relations are shown as follows (the arguments and irrelevant C statements are omitted).

```c
int main(void)
{
    ...
    func1(...);
    func2(...);
    ...
}
```

```c
int func1(...)
{
    ...
    func1(...);
    ...
}
int func2(...)
{
    ...
    func3(...);
    func4(...);
    ...
}
```
Sample PARTs in Question 2 (Cont.)

func1() is a recursive function and calls itself 15 times for the actual parameters given in main(). Both func3() and func4() do not call any function. The sizes of all stack frames are shown as follows.

- main(): 200 bytes.
- func1(): 100 bytes.
- func2(): 400 bytes.
- func3(): 1,400 bytes
- func4(): 300 bytes

How much stack space is needed to execute this program correctly? (3 marks)

Sample PARTs in Question 3

Write an AVR assembly program to find the maximum value of all array elements in the one-dimensional integer array A. Your program must satisfy the following requirements.
- Array A has 10 elements and each element is a 2-byte signed integer.
- Each element A[i] (i=0, 1, ..., 9) has an initial value that is stored in the flash memory. You may choose any integer value for each element.
- Array A is stored contiguously in the SRAM.
- Your program must define and use at least one MACRO.
- The maximum value found by your program is stored in registers r26:r25. (10 marks)

Sample PARTs in Question 3 (Cont.)

Write an AVR assembly program to implement the following C program.

```c
int sum(int n);
int main(void) {
    int n=100;
    sum(n);
    return 0;
}
```

int sum(int n)
{
    if (n<=0) return 0;
    else return (n+ sum(n-1));
}

All local variables and parameters must be stored in the stack space. You need to choose a proper size for n and describe the stack frame structure using a diagram. (10 marks)
Sample PARTs in Question 4

Assume for a processor with a 700 MHz clock it takes 150 clock cycles for a polling operation (calling polling routine, accessing the device, and returning). The overhead for an interrupt operation is 200 clock cycles. Hard disk transfers data in 128-byte chunks and can transfer at 6 M bytes/second rate.

If the processor uses software polling, what percentage of the processor time is tied up in polling the hard disk to achieve a data transfer rate of 6M bytes/second? (4 marks)

Sample PARTs in Question 4 (Cont.)

If the processor uses interrupt technique and the interrupt rate is equal to software polling rate, what percentage of the processor time is tied up in servicing interrupt by the hard disk during the data transfer? (4 marks)

Sample PARTs in Question 4 (Cont.)

Consider an embedded system using an AVR mega64 microcontroller. There are 5 interrupting sources, Int0, Int1, …, Int4. The size of the stack space needed by each interrupt service routine, denoted also by Inti (i=0, 1, …, 4) for simplicity, is shown as follows.

Int0: 100 bytes
Int1: 50 bytes
Int2: 20 bytes
Int3: 40 bytes
Int4: 50 bytes

If no interrupt occurs, the correct execution of the program needs 1K bytes of stack space in the worst case.

Sample PARTs in Question 4 (Cont.)

If interrupts may occur at any time and each interrupt service routine does not enable the Global Interrupt Flag I in the Program Status Register, how much stack space is needed by the program in the worst case? (4 marks)