### You can download from File>Download As

### 1. Convert the following numbers from the original base to the specified base:

#### a) 12310-> \_\_\_\_\_\_\_\_\_\_\_\_\_\_2

#### b) 101012 -> \_\_\_\_\_\_\_\_\_\_\_\_\_\_10

#### c) 108410 -> \_\_\_\_\_\_\_\_\_\_\_\_\_\_16

#### d) A516 -> \_\_\_\_\_\_\_\_\_\_\_\_\_\_10

#### e) 110010012 ->\_\_\_\_\_\_\_\_\_\_\_\_\_\_16

#### f) 2D516 -> \_\_\_\_\_\_\_\_\_\_\_\_\_\_2

### 2. What is the result of the following calculations?

**a)** 1395 + 4988 (base 16)

**b)** 11001001 + 00101101 (base 2)

**c)** A41 – 560 (base 16)

**d)** 11001 – 011 (base 2)

### 4. What number does 10010010 represent as an unsigned number? What does it represent in 2’s complement notation?

### 5. In 2’s complement addition, 11011011 + 01100000 = 00111011. Was there a 2’s complement overflow? Why? What do the values in this sum represent?

### 6. What is the difference between performing 2’s complement addition and unsigned addition in the AVR processor?

### 7. Represent the following numbers in IEEE 754 32-bit floating point notation:

#### a) 1.5

#### b) 1084

#### c) –1

#### d) –13.75

### 8. What does the following IEEE 754 FP number represent:

#### 0 1000 0001 110 0000 0000 0000 0000 0000

#### Sign Exponent Mantissa

### 9. Encode the following instructions into Atmel AVR machine code:

#### a) ldi r18, 127

#### b) mov r18, r2

#### c) lds r2, 0xABCD

### 10. How many bits are needed to address:

#### a) 16 32-bit general purpose registers?

#### b) a memory space of 65536 bytes (assume byte addressing)?

#### c) a memory space of 65536 32-bit words (assume byte addressing)?

### 11. What do the following letters in a typical status register stand for and how are they generated?

#### a) Z

#### b) C

#### c) V

#### d) N

#### e) S

### 12. What is the main difference between the memory models of Princeton (von Neumann) and Harvard architectures?

### 13. Based on the below, what is the 32-bit word stored at the memory address 0x00000100 in the options below

Memory address Data

0x00000100 0xAF

0x00000101 0x1B

0x00000102 0xC2

0x00000103 0x05

#### a) big-endian machine?

#### b) little-endian machine? (Starts with least significant byte @ smallest address)

### 14. Can you design an 8-bit instruction format that can allow 4 2-operand instructions for a machine with 8 registers?

### 15. What do these notations mean in AVR assembly programming? Where are they used?

**a) .def d) .dseg g) .dw**

a. .def:

d. .dseg:

g. .dw:

**b) .set e) .org h) .byte**

b. .set:

e. .org:

h. .byte:

**c) .cseg f) .db i) .equ**

c. .cseg:

f. .db:

i. .equ:

### 16. Where are the functions low() and high() utilised? Load -200 into a two byte number.

### 17. What are the differences between Macros and Functions? In what circumstances are each of them appropriate, and when should each be avoided? Write a Macro called Invert to invert the value of a register

### 18. What are word addressable and byte addressable? Explain them with examples using AVR memories.

### 19. Consider the following AVR assembly code segment and fill the initialization part?

**.dseg**

**array: .byte 20**

**.cseg**

**data: .dw 0x1234**

**// Initialize the X pointer with array**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**// Initialize the Z pointer with data**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### 20. What are little endian and big endian representations ? Which endian is used in AVR?

### 

### 21. Identify the errors in the following instructions,

#### a) ldi r1,18

#### b) cp r16,‘L’

#### c) ldi zh, high(0x3476) => Word Addressable

#### d) ldi r40, 23

#### e) brge loop => for both unsigned numbers

#### f) brlo end => for both signed numbers

### 22. Write AVR assembly code segments for the following scenarios,

#### a) Initialize an array A of size 20 (each element is one byte) with values ranging from 1 to 20.

#### b) Initialize an array B of size 20 (each element is two bytes) with values ranging from -1 to -20.

#### c) Add the arrays A and B together and store the result into an array C.

#### d) Store the string 12345678 into program memory using .db and .dw.

#### e) Load the values stored in the program memory in (d) and store them into data memory in the reverse order.

### 23. How do you multiply a two byte number by a one byte number? (Explain using a simple example). Do we have to consider the carry bit in the STATUS register for this case?

### 24. Investigate the different ways of writing AVR assembly code for the following scenarios

#### a) Copying a pair of registers into another pair of register.

#### b) Multiply a number by 4.

#### c) Divide a number by 4.

### 25. When are MUL, MULS and MULSU instructions used and how are they are used?

### Write AVR assembly code to perform multiplication for the following set of numbers,

#### a) 10, 12 (1 byte result)

#### b) -11,11 (1 byte result)

#### c) -4,-14 (1 byte result)

#### d) 32,258 (2 bytes result)

#### e) -352, 28 (2 bytes result)

#### f) -27,-375 (2 byte result)

### 26. 1 Minimally modify the code below to add two numbers (in r17:r16 and r19:r18) when the result is bigger than 255.

**ldi r16, 1**

**ldi r17, 0**

**ldi r18, 255**

**ldi r19, 0**

**add r16, r18**

**add r17, r19**

### 26.2 Write AVR code to add two 32 bits values?(Using R16-R23 to hold all values.)

**a = 0x00000100**

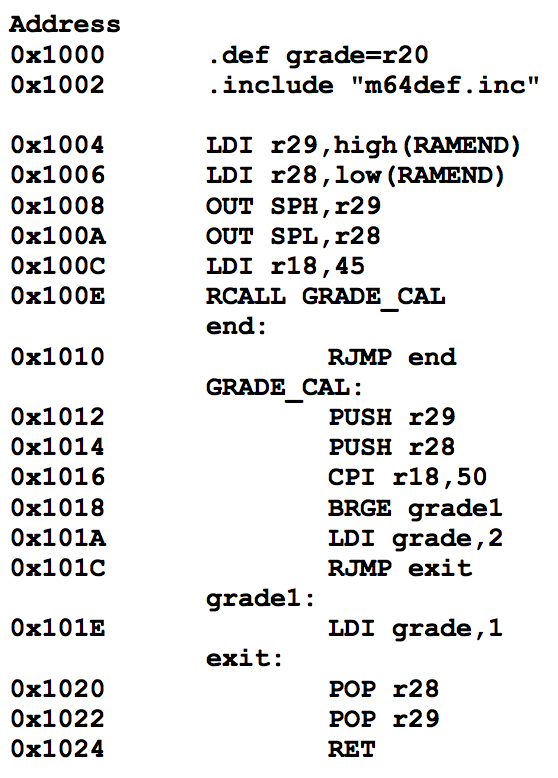
**b = 0x002000FF**

### 27. Please complete the following table with instructions used for each operation.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Instructions** | **Registers** | **Stack** | **Data Memory** | **Program Memory** | **Separate IO** | **Memory-Mapped IO** |
| **Initialize** |  |  |  |  |  |  |
| **Write to** |  |  |  |  |  |  |
| **Read from** |  |  |  |  |  |  |

### 28. How do you setup a port to act as an input port or as an output port in AVR? What instructions are used to read from an I/O port? What instructions are used to write to an I/O port?

### 29. Consider the following example AVR code segment:



#### What are the values of r28, r29, SPL and SPH:

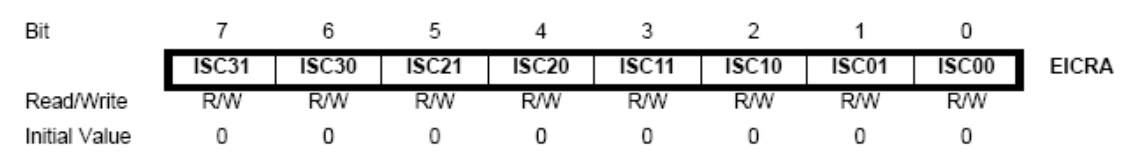
#### a) after line “LDI r28,low(RAMEND)”?

#### b) after line “OUT SPL,r28”?

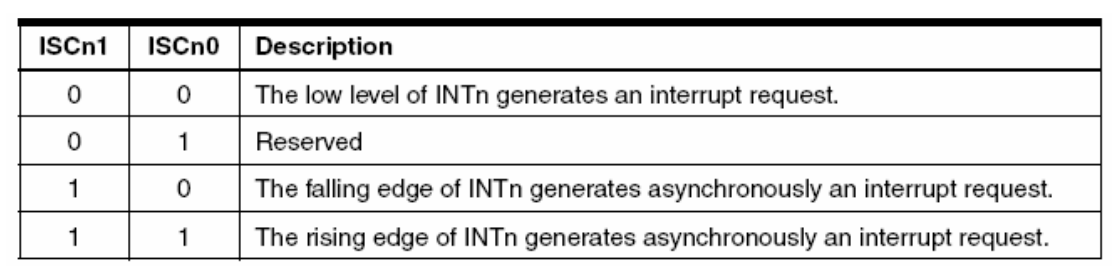
#### c) after line “BRGE grade1”?

#### d) after line “POP r29”?

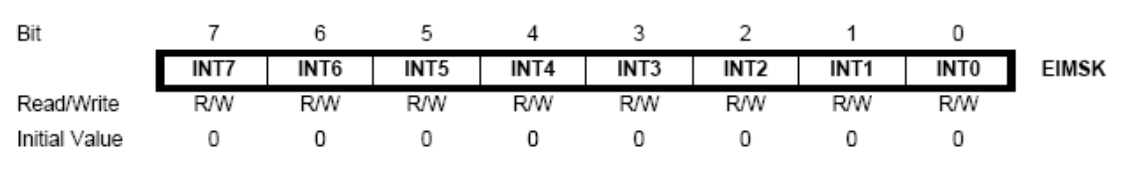
### 30. The EICRA register is used to indicate what condition should be present for external interrupts to occur, and looks like this:



**where each pair of bits ISCn1 and ISCn0 mean the following for INTn:**



**The EIMSK register is used to enable the external interrupts and looks like this:**



**In “m64def.inc”, the values in these registers have been defined to their bit value. e.g., ISC00 = 0, ISC11=3 and INT2=2. Knowing this, examine the following code:**

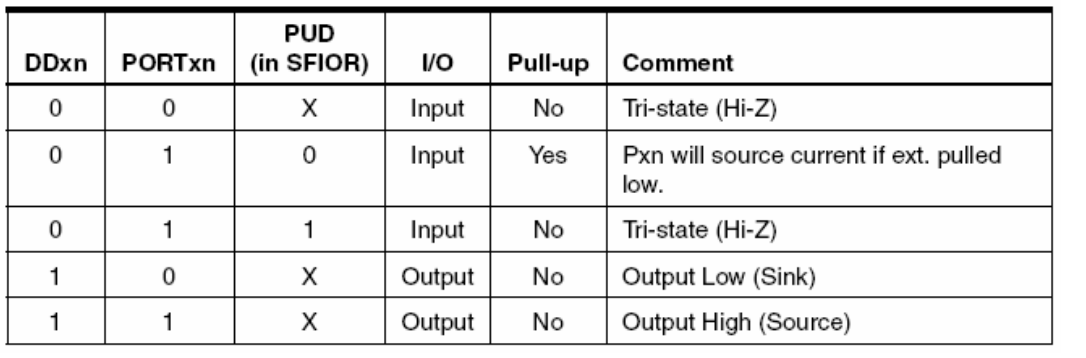
#### Screen Shot 2014-06-25 at 6.03.25 PM.png. a) What is the value (in binary) that is written to the EICRA register?

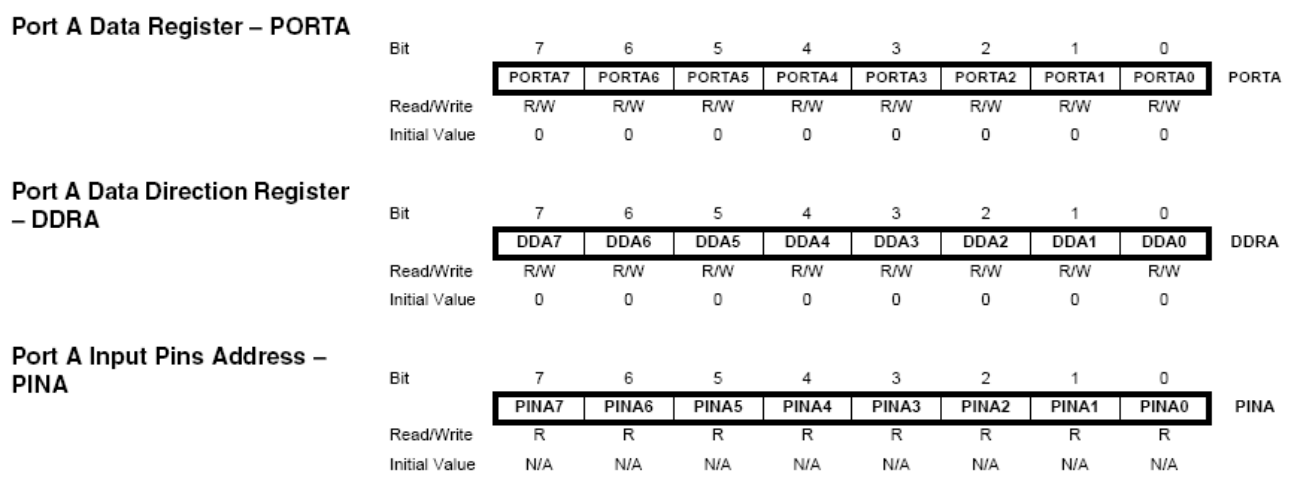
#### b) Why do we use this approach to set up the register values?

#### c) Which external interrupts can occur, and when will they occur?

#### d) What is the difference between the ‘sts’ instruction and the ‘out’ instruction?

### 31. This question looks at the registers associated with PORT A. The following tables might help:





#### a) What is the purpose of the DDRA register?

#### b) What is the purpose of PORTA0 when DDA0 = 1?

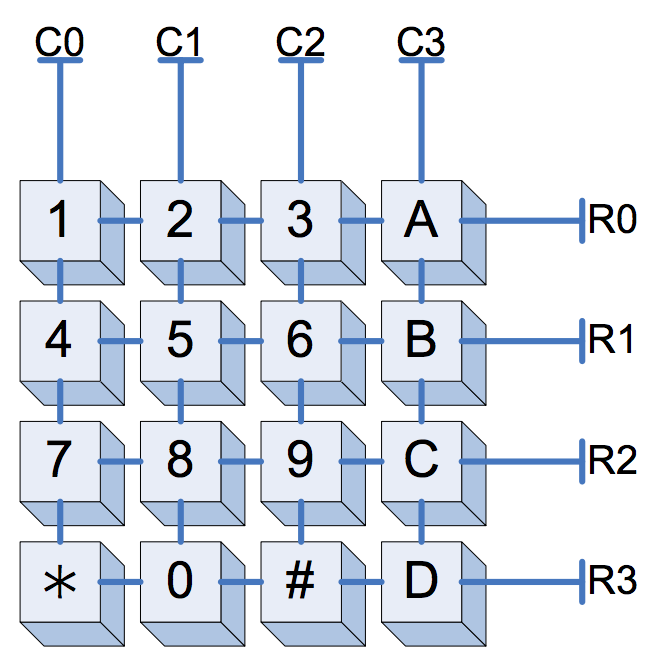
#### c) What is the purpose of PORTA0 when DDA0 = 0 and PUD = 0?

#### d) What is the purpose of the PINA register?

### 

### 

### 32. The Keypad on the AVR boards is a set of 16 push buttons. The keypad has four rows and four columns, accessible via the pins R0-R3 and C0-C3. When you push a button on the keypad, it connects the column of the key to the row of the key as follows:



One method to correctly read what keys are being pressed is to:

1: Set up the rows so that they read a Logic 1 when none of the buttons on the row is pushed.

2: Set one column to Logic 0 and all other columns to Logic 1.

3: Read the values of the row pins. If a row reads as Logic 0, you know that the switch at that row and column must be pushed.

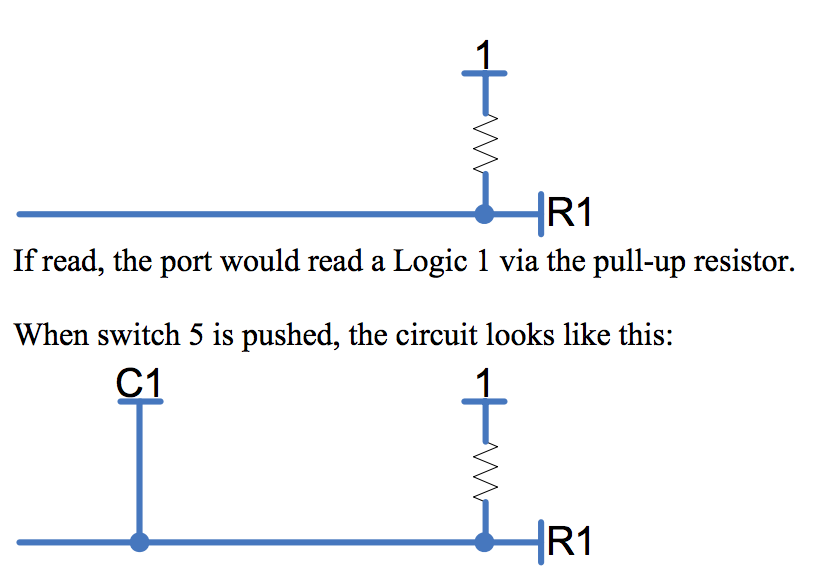
4: Set a different column to Logic 0 and read the rows.

5: Repeat steps 3 and 4 until a switch is found to be pressed or you run out of columns.

6: Repeat steps 2-5 again if you want to see whether a different switch is pushed.

Part of your third lab requires you to perform this algorithm. Steps 2-5 should be fairly simple to code, but step 1 is not so obvious. The way to accomplish this is with pull-up resistors. Pull-up resistor ties an input pin to Logic 1 via a resistor. This means that an input pin will still read any value that is input, and will read Logic 1 if disconnected.

To further understand this, look at switch 5 in the above diagram. When none of the switches connected to row 1 are pushed, the circuit (with pull-up shown) looks like this:



In this case, the port connected to R1 will always read the current value of C1. When C1 is Logic 0 there will be a voltage drop across the resistor, but this will not affect the value being read. Thus, the pull-up resistor accomplishes the desired task.

#### a) How do you setup an AVR I/O port so that it has pull-up resistors connected to its input pins? (See question 2 of this tute)

#### b) Write the code to find a switch that has been pushed by scanning either the columns or rows. (You have to do this for your lab, anyway)

#### c) Can you see an electrical problem with this scanning method when two switches on the same row are pushed at the same time (e.g., 5 and 6)? How could you correct this?

#### (Hint: There might be something better you can do than output logic 1s to the columns you are not testing.)

#### 

# Extra Qns (by Oliver Tan)

# Numbers Questions

<https://docs.google.com/document/d/1i-_rCVuaJfo4biF-G9PttJkhioAO8T9bUFw8kZbX32I/edit>

1. Which of the following rows have equivalent values:

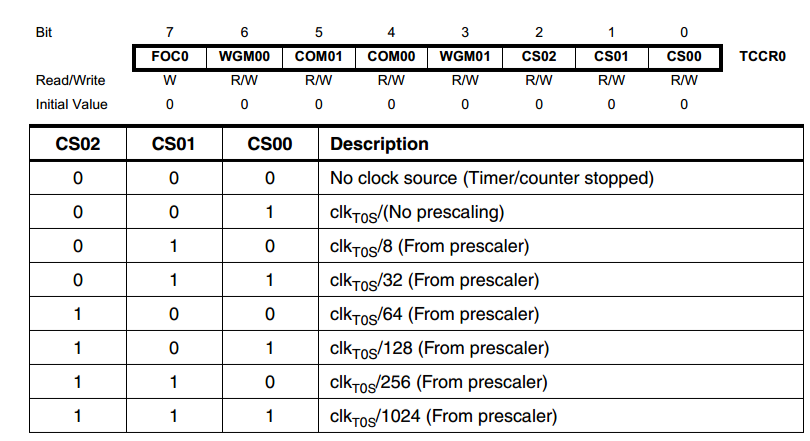
|  |  |  |  |
| --- | --- | --- | --- |
| (a) | -7 | 000001112 | -0x7 |
| (b) | 256 | 0xF | 111111112 |
| (c) | 1000000002 | 256 | 0xF |
| (d) | -2 | 0xFE | 111111102 |
| (e) | 0x1 | 256 | 111111112 |

1. What is the binary result of 100111002 - 11110012 if it is an eight bit operation.
   1. 001000102
   2. 001000112
   3. 010100102
   4. 010100112
   5. 1001000112
2. Which of the following statements is correct:
   1. When performing operations on four bit numbers, to store the result, I need at most 8 digits for addition and 8 digits for multiplication.
   2. When performing operations on four bit numbers, to store the result, I need at most 4 digits for addition and 4 digits for multiplication.
   3. When performing operations on four bit numbers, to store the result, I need at most 5 digits for addition and 8 digits for multiplication.
   4. When performing operations on four bit numbers, to store the result, I need at most 4 digits for addition and 8 digits for multiplication.
   5. When performing operations on four bit numbers, to store the result, I need at most 5 digits for addition and 16 digits for multiplication.
3. What is the hexadecimal representation of the result of adding 0x40D00000 and 0x40080000 if these numbers were of the IEEE 754 Single-Precision Floating Point standard:  
     
     
   Start with the hexadecimal values...  
   40D00000  
   40080000  
   Convert to binary and split into floating point formatting...  
   0 1000 0001 1010 000 0000 0000 0000 0000  
   0 1000 0000 0001 000 0000 0000 0000 0000  
   Convert to regular binary value and add...  
    110.1  
   + 10.001  
   -------------  
   1000.101  
   Convert sum back to floating point formatting...  
   0 1000 0010 0001 0100 000 0000 0000 0000  
   Convert to hexadecimal value...  
   410A0000  
   1. 0x41C50000
   2. 0x410A0000
   3. 0x41A50000
   4. 0x42528000
   5. 0x40940000

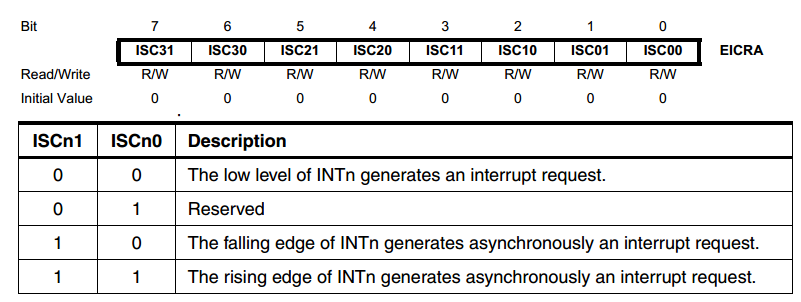
# Interrupt Questions

<https://docs.google.com/document/d/1ZFA1ddqZG8lig_eTLmQa0_Sd7V7OtoFDdP3aQxODTKs/edit>

1. Which one of these cannot explicitly trigger an interrupt?
   1. Watchdog Timers
   2. Board Reset
   3. Keypad Pressing? (polling keypad instead of interrupts)
   4. CTRL+C on Ubuntu
   5. Playing Music?
2. Which of the following is NOT on the stack after an interrupt preamble (on the AVR)?
   1. a copy of the return address
   2. a copy of the status register(s)
   3. a copy of the conflict registers
   4. variables that are used in the previous function?
   5. none of the above
3. What does the interrupt do on the software side to activate the appropriate operations on the AVR board?
   1. The PC will change directly into the appropriate instruction, which is at a hardcoded location
   2. The program will automatically lookup “INT\_0\_HANDLER” in the code and the PC will jump to this location
   3. The PC will jump to the address indicated by the contents of 0x0002
   4. The PC will jump to the base of the function, which is determined by finding the number of iterations of “reti”
   5. The PC will jump based on a interrupt vector table



1. Given the image above, how would I load the temp variable so that when it outputs to TCCR0, the prescaler is set to 8?
   1. ldi temp, 010
   2. ori temp, (10 << CS01)
   3. ldi temp, 8
   4. ldi temp, (10 << CS00)
   5. ldi temp, (2 << CS00)



1. How would you load the “temp” register to, considering the interrupts which may have already been initialised, create an interrupt on the falling edge of INT3 when the temp register is loaded into EICRA?
   1. ldi temp, (1 << ISC30)
   2. ori temp, (1 << ISC30)
   3. ori temp, (10 << ISC30)
   4. ldi temp, (1 << ISC31)
   5. ori temp, (1 << ISC31)

# SAMPLE FINAL EXAM 2

<http://www.cse.unsw.edu.au/~cs2121/LectureNotes/wk14.pdf>

## 1. Basics Concepts (12 × 3 = 36 marks)

### PART 1 Describe the difference(s) between microprocessors and microcontrollers. What is ATmega64?

### PART 2 Explain the concept of endianness. Which endianness does AVR use? Please give an example.

### PART 3 Function can be recursive. Can macro be recursive? Why?

### PART 4 Describe differences between memory-mapped I/O and separate I/O.

### PART 5 In ADC, what is resolution? And what is accuracy? What are the differences between these two terms? Please give an example.

### PART 6 What is stack frame? Draw a memory map to show the basic structure of a stack frame. Please list instructions that can access stack in AVR.

### PART 7 What is watchdog timer? What should be done to set up watchdog timer before to use it?

### PART 8 What is aliasing in ADC? How to avoid aliasing?

### PART 9 What does USART stand for? In asynchronous serial communication, how is receiver clock is synchronized to a transmission operation of the transmitter?

### PART 10 What is interrupt vector table? How many interrupts are available in Atmega64? The following is the part of interrupt vector table in an AVR program. Is this part of vector table correctly set? Why?

### ; interrupt vector table

### .org 0x0000

### rjmp RESET

### rjmp INT0

### rjmp INT1

### PART 11 There is no software interrupt available in AVR. How can you implement a software interrupt in AVR?

### PART 12 The keypad is a typical input device in microcontroller application. Write a high level description (algorithm) that specifies how the input data from keypad is obtained by the microcontroller.

## 2. Miscellaneous Questions (31 marks)

### PART 2 Consider the content of the AVR program memory in hex format as shown below. (a) What sequence of 4 ASCII characters does the content correspond to?

### (b) What 2-byte signed integer values in base 10 does the content correspond to?

(c) What 2-byte unsigned integer values in base 10 does the content correspond to? (6 marks)



0x3C4E = 1543810

0x535A = 2133810

### PART 3 Consider the following AVR assembly code:

**.MACRO delay**

**loop: subi @0, 1**

**sbci @1, 0**

**nop**

**nop**

**nop**

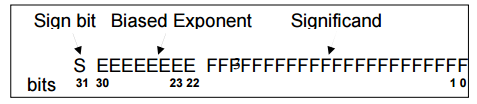
**nop**

**brne loop ; taken branch takes two cycles.**

**.ENDMACRO**

### All instructions in the program are 2 bytes long. What is the size of the code in bytes? How many parameters does the macro have? What is the range of values of each parameter? The code can generate a delay. What is the range of the delay? Assume the processor frequency is 8 Mhz. (7 marks)

### PART 4 Consider two single precision floating point numbers x and y in IEEE 754 format, where x= 0x50240000 and y=0x40080000. The IEEE 754 format is given as follows:



### What is the decimal value of x+y? Please show your work. (4 marks)

### PART 5 How many bits do you need to represent a~z 26 letters and 0~9 ten digits? Can you encode them with the 4x4 keypad ? If no, why? If yes, how? (5 marks).

### 

## 3. AVR Assembly Programming and Design (33 marks)

### PART 1 Write an AVR assembly program to find the max value of an integer array, A. Your program must satisfy the following requirements.

### 1) Each element is a 2-byte signed integer.

### 2) Array A is stored contiguously in the FLASH.

### 3) Your program must define and use at least one MACRO.

### 4) The array length is 10. (7 marks)

**PART 2 An array of ten 2-byte integers are stored in the AVR program memory. Write a program to convert the array to different endianness format and store it back to the data memory. 5 Your program must satisfy the following requirements:**

### 1. Your program must use at least one function.

### 2. All local variables and parameters must be stored in the stack space.

### 3. You must describe the stack frame structures for the function used in your program. (8 marks)

### PART 3 Consider to design an embedded system to control the speed of a DC motor. The operating specification of the system is given below:

### 1. The speed of the motor is input from the keypad

**i. Assume that the motor is driven by a PWM signal. And there is a formula to determine the duty cycle for different motor speed when the motor spins without any extra load.**

### 2. The motor is started by an external push button operation

### 3. The speed deviation must be controlled within a specified limit.

### 4. If the speed deviation exceeds the limit, the LED bar is set to ON within 1 second to alarm the user and at the same time the motor is stopped.

### Your design should include:

### 1. definition of all tasks in your system.

### 2. task scheduling diagram that specifies the execution timing frame for each task

### 3. interrupt design that includes what kind of interrupts you are going to use and the purpose for using each of the interrupts

### 4. a code template that includes interrupt handling subroutines

### Note, the code template does not have to be complete. You can use comment lines for sections of code that involve detail setting information and can be inserted later. Examples are given below.

### ; insert code here to set up timer0 for 1 second timer out

### ; insert code here to enable timer0 overflow interrupt

### (18 marks)

#### 