

ELEC1601_ELEC9601 Main Final Exam

ⓘ This is a preview of the published version of the quiz.

Started: Nov 10 at 20:49

Quiz Instructions

Your exam comprises 19 questions. The first 16 questions are worth 32 points in total and the final 3 questions are worth 18 points.

Question 17 requires that you provide your answers in the quiz and **upload your working outs to the question to the file upload assignment**

(<https://canvas.sydney.edu.au/courses/37583/assignments/336505>) after you have submitted your quiz.

You have an additional 15 minutes to complete this task. Do not treat this as additional writing time. Non-submission owing to running out of time is not considered a technical issue.

Question 1

2 pts

A machine code must support arithmetic instructions of the following format:

opcode, destination register, source register 1, source register 2.

e.g. ADD, R0, R1, R2 translates to add $R1 + R2$ and store the result in R0

The total number of different arithmetic instructions that must be supported is 199, with 2^{30} different registers. All bits for this instruction are utilised in the encoding.

What is the minimum number of bits required for this machine code?

Question 2

2 pts

Suppose a computer system with a Harvard architecture is able to store 38,597 instructions and 814,498 data values. Each instruction and data value is 32 bits and are byte addressed.

What is the minimum number of address bits required for the memory where data values are stored?

Question 3

2 pts

Complete the truth table for the following boolean function:

$$(A + B')(B + C') + AB$$

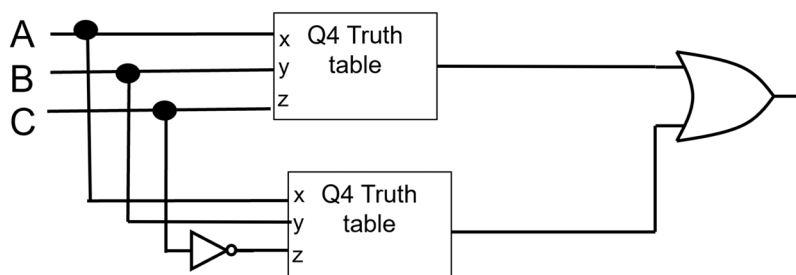
A	B	C	out
0	0	0	<input type="text"/>
0	0	1	<input type="text"/>
0	1	0	<input type="text"/>
0	1	1	<input type="text"/>
1	0	0	<input type="text"/>
1	0	1	<input type="text"/>
1	1	0	<input type="text"/>
1	1	1	<input type="text"/>

Question 4**2 pts**

Suppose you have a circuit that implements the following truth table:

x	y	z	Out
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

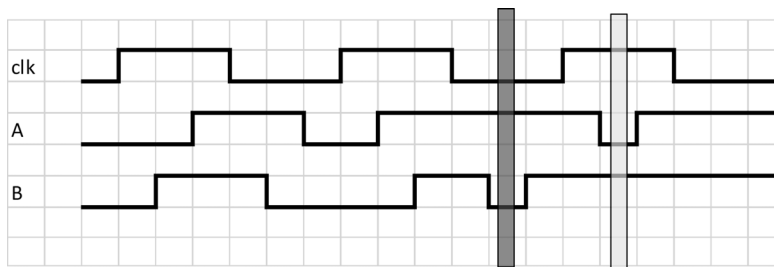
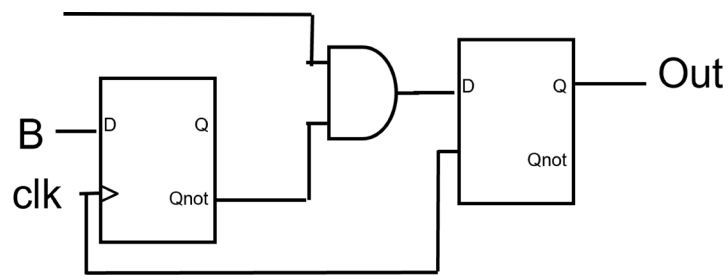
Suppose you then create two instances of this circuit and wire them up as follows (the second instance has its final input inverted):



What is the output of the circuit if a=1, b=0, c=0?

Question 5**2 pts**

Study the following circuit and timing diagram (consisting of D-type latches, D-type Flip-Flops and Logic Gates):



What is the value for Out at the highlighted times (if it cannot be determined, write UNKNOWN)

(highlighted red) =

(highlighted yellow) =

Question 6

2 pts

Assume that a memory is initialised as follows:

Memory Address	Cell Contents
0xC336	0x00
0xC337	0x01
0xC338	0x01
0xC339	0x11

0xC33A	0x10
0xC33B	0x10

Suppose the X register is initialised to 0xC338 and the following commands are issued:

LD R1, X+

LD R2, X+

ADD R1, R2

ST X, R1

What is the value in 0xC338? (Write your answer in decimal)

Question 7

2 pts

Suppose you have a computer system with a 6 stage pipeline and clock period of 166 ns.

Assuming there are no pipeline stalls (no branching, no I/O requests, no interrupts etc.), how long would it take to execute the following instructions:

ADD R1 R2

ADD R1 R3

ADD R1 R4

ADD R1 R6

Registers R1 to R6 are initialised to 5, 4, 8, 2, 5 and 4 respectively

Question 8**2 pts**

Assume that a memory is initialised as follows:

Memory Address	Cell Contents
0x00C336	0x23
0x00C337	0x61
0x00C338	0x63
0x00C339	0x89
0x00C33A	0x25
0x00C33B	0x69

A computer system has the value 0x00C338 in its stack pointer. The stack grows (when you push data) towards lower memory positions. Suppose the system executes the sequence of instructions:

- POP R1
- POP R2
- ADD R1 R2
- PUSH R1
- PUSH R2
- PUSH R3

What value is in the memory location 0x00C338. (Write your answer in decimal. If it cannot be computed given the above information, enter the value 0)

Question 9**2 pts**

Suppose the following two instructions are executed:

SUBI R20, 7

BREQ Destination

The machine code for the BREQ instruction is as follows:

1111 0000 0001 1001

Assume the following register values:

Program counter = 0x101

R20 = 0x010

What is the new value of the PC?

The relevant information for the instruction set is given below:

18. BREQ – Branch if Equal

18.1. Description

Conditional relative branch. Tests the Zero Flag (Z) and branches relatively to PC if Z is set. If the instruction is executed immediately after any of the instructions CP, CPI, SUB, or SUBI, the branch will occur if and only if the unsigned or signed binary number represented in Rd was equal to the unsigned or signed binary number represented in Rr. This instruction branches relatively to PC in either direction ($PC - 63 \leq \text{destination} \leq PC + 64$). Parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 1,k.)

Operation:

- (i) If $Rd = Rr$ ($Z = 1$) then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Syntax:

Operands:

Program Counter:

- (i) BREQ k

$-64 \leq k \leq +63$

$PC \leftarrow PC + k + 1$

$PC \leftarrow PC + 1$, if condition is false

16-bit Opcode:

1111	00kk	kkkk	k001
------	------	------	------

124. SUBI – Subtract Immediate

124.1. Description

Subtracts a register and a constant, and places the result in the destination register Rd. This instruction is working on Register R16 to R31 and is very well suited for operations on the X, Y, and Z-pointers.

Operation:

$$(i) \quad R_d \leftarrow R_d - K$$

Syntax:

Operands:

Program Counter:

$$(i) \quad \text{SUBI } R_d, K$$

$$16 \leq d \leq 31, 0 \leq K \leq 255$$

$$PC \leftarrow PC + 1$$

16-bit Opcode:

0101	KKKK	dddd	KKKK
------	------	------	------

124.2. Status Register and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow

$$\mathbf{H} \quad \overline{R_d3} \cdot K3 + K3 \cdot R3 + R3 \cdot \overline{R_d3}$$

Set if there was a borrow from bit 3; cleared otherwise.

$$\mathbf{S} \quad N \oplus V, \text{ for signed tests.}$$

$$\mathbf{V} \quad R_d7 \cdot \overline{K7} \cdot \overline{R7} + \overline{R_d7} \cdot K7 \cdot R7$$

Set if two's complement overflow resulted from the operation; cleared otherwise.

$$\mathbf{N} \quad R7$$

Set if MSB of the result is set; cleared otherwise.

$$\mathbf{Z} \quad \overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$$

Set if the result is \$00; cleared otherwise.

Question 10

2 pts

Study the following code (you can assume ". . ." refers to code that is not shown)

```
main:
    ...
    LDI R27, hi8(d1)    ; PC = 0x0A38, SP = 0x0B13
    LDI R26, lo8(d1)
    LD  R18, X+
    ADD R18, R18
    PUSH R0
    CALL subroutine1
    POP R18             ; What is the value of the SP at this line of code?
    ...

subroutine1:
    POP R18
```


...
RET

You can assume that the address of subroutine1 is 0x1A19 and that this code was generated by the AVR-GCC compiler and follows the relevant conventions

What is the value of the Stack Pointer at the highlighted line of code? (Write your answer in Decimal)

Question 11

2 pts

Consider the following definitions for d1, which represents a 4 byte array of integer values:

- d1: .byte 0, 1, 2, 3

(1st element of the array for d1 has the value 0, 4th element of the array has the value 3)

If d1 is located in address 0x0534, what is the value of the second element of d1 after executing the following instructions? (Write your answer in Decimal)

- LDI R27, hi8(d1)
- LDI R26, lo8(d1)
- LDI R19, hi8(d1)
- LDI R18, lo8(d1)
- ST X+, R18
- ST X+, R19

Question 12**2 pts**

An AVR assembly program defines the following variables and labels:

```
.section .data
```

```
D1: .byte 1, 4, 3
```

```
D2: .byte 9, 5, 2
```

If the address of D1 is 0x57, what is the address of D2? (Write your answer in Decimal)

Question 13**2 pts**

What is the decimal value (base 10) held in R9 after the following sequence of instructions?

```
LDI R18, 0x229
```

```
MOV R9, R18
```

```
ADD R9, R9
```

Question 14**2 pts**

Study the following program

It was generated by the compiler avr-gcc that uses the AVR libc library, so obeys the convention for register management

```
LDI R18, 9
PUSH R20
PUSH R18
CALL subroutine1; Call the subroutine
POP R2
POP R2
ADD R20 R2
ADD R20 R18
...
```

```
subroutine1:
...
; Values for the registers after this line of code are shown in the text below
PUSH R0;
RET;
```

Suppose at the end of the subroutine (the line highlighted in Red) R0=3, R2=5, R18=6

What is the value stored in R20 after the line highlighted in Green is complete? Write 0 if unknown. Write your answer in Decimal

Question 15

2 pts

Study the following program

It was generated by the compiler avr-gcc that uses the AVR libc library, so obeys the convention for register management

```
LDI R20, 6
LDI R21, 7
LDI R22, 7
LDI R23, 7
LDI R24, [a5]
PUSH R20
PUSH R21
PUSH R22
PUSH R23
```

```
PUSH R24
CALL subroutine1; Call the subroutine
...
```

```
subroutine1:
    IN R31, 0x3E    ; Z <- SP
    IN R30, 0x3D
    LDD R18, Z+4
    LDD R19, Z+5
    ADD R18, R19    ; What is the value for R18 after execution of this line?
    ...
```

Reminder, the IN R31, 0x3E ; IN R30, 0x3D commands load the stack pointer into the Z register

What is the value stored in R18 in subroutine 1 (before the ...). Write 0 if unknown.
Write your answer in Decimal

Question 16

2 pts

Study the following handwritten (potentially buggy) code (you can assume "... " refers to code that is not shown

```
main:
    ...
    LDS R25, x
    PUSH R25,
    CALL subroutine1 ; This instruction is at location 0x1285
    POP R0,
    ...
subroutine1:                ; This subroutine is at location 0xAB23
    ...
    MOV R25, R8
    PUSH R0
    CALL subroutine2
    POP R25
    ...
    MOV R24, R7    ; result
    RET
subroutine2:                ; This subroutine is at location 0xAD75. No code is missing from
this subroutine
    POP R0
    ADD R0 R0
    PUSH R0
    RET
```

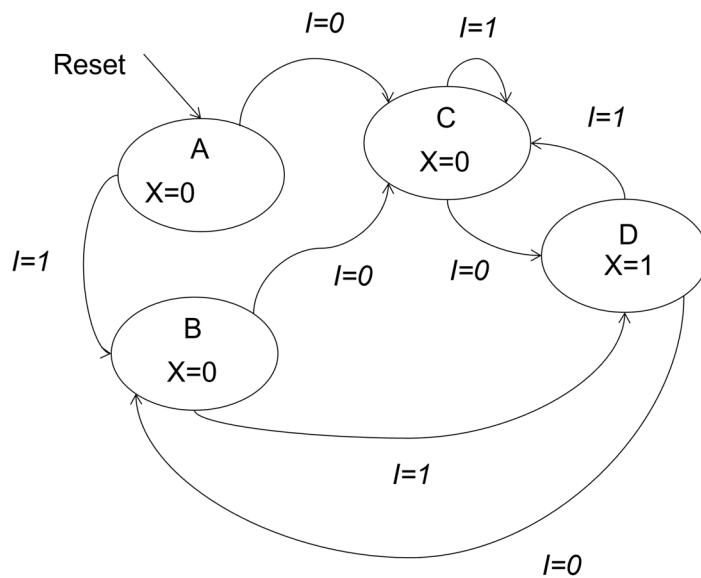
Assume the return address stored in the stack by the CALL instruction is 3 bytes.

What is the value of the Program Counter after the RET instruction in Subroutine 2?
(Write your answer in Decimal, write 0 if unknown)

Question 17

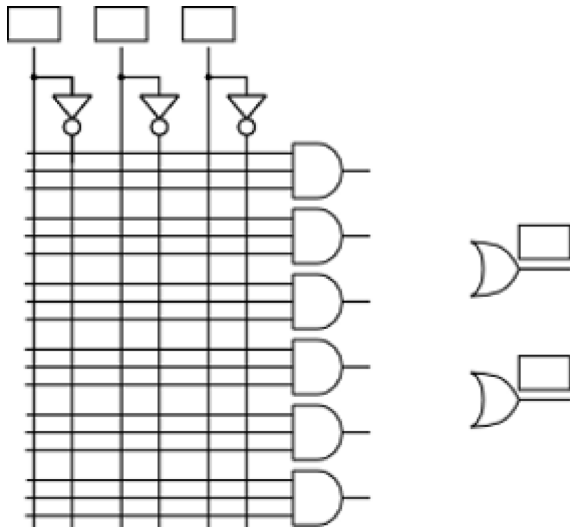
0 pts

Study the following Finite State Machine:



First create a truth table showing how to compute the next state from the current state. (3 Marks)

Secondly, show how you could create a combinational digital logic circuit that computes the next state based on the current state. You should draw your answer on a PLA style device, as shown below: (3 Marks)



You may add extra AND gates as is necessary.

Draw your answer, take a picture and upload it afterwards to the [file upload assignment](https://canvas.sydney.edu.au/courses/37583/assignments/336505) (<https://canvas.sydney.edu.au/courses/37583/assignments/336505>) .

Please type 1 in the following box to show you understood this instruction [a17]

Edit View Insert Format Tools Table

12pt Paragraph | **B** *I* U A √ √ T² √ |

√ √ √ | √ | √ √ √ | √

p

| 0 words |

Question 18

6 pts

Consider the following fragment of code written in a high level programming language:

```
int cost[5] = {1, 2, 3, 4, 5};
int total_cost = 0;
int count = 0;
int temp;
temp = roll_dice();
while ((temp > 3) || count < 5)
{
    temp = roll_dice();
    total_cost = total_cost + cost[count];
    count++;
}
```

Write a sequence of AVR assembly instructions equivalent to the previous code. You can assume the function `roll_dice()` is written elsewhere. You do not need to write this. However, your assembly code must include call this subroutine and assume the result from the subroutine is returned via the stack.

You can assume the data section is defined as follows:

```
.segment .data
data: .byte 1, 2, 3, 4, 5
result: .space 1, 0
count: .byte 0
```

Edit View Insert Format Tools Table

12pt ▾ Paragraph ▾ | **B** *I* U A ▾  ▾ T² ▾ |

 ▾  ▾  ▾ |  ▾ |  ▾  ▾  ▾ |   ▾  

p



0 words

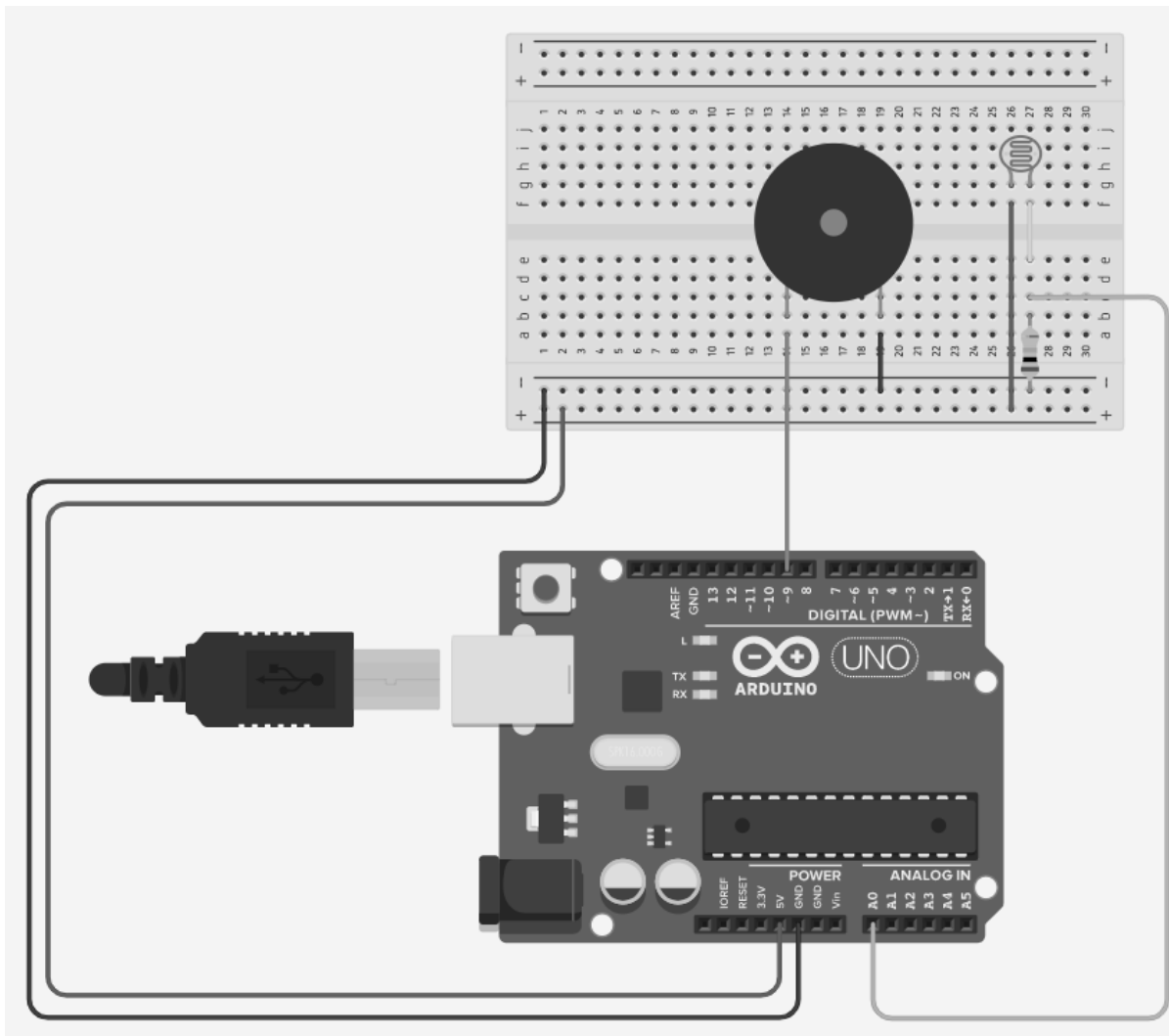
</>



Question 19

6 pts

Study the following TinkerCad diagram:




This monitors a light sensor connect to Analog Pin A0 and Buzzer connected to digital PIN 9.












Create a program that checks the value of the light sensor every second. If the light intensity increases, increase the pitch of the Buzzer by 10 Hz up to a maximum of 1000Hz. If the light intensity decreases, reduce the pitch to a minimum of 500 Hz.

You may make use of the following template:

```
// Global variable declarations
.....
.....
void setup()
{
.....
.....
}
void loop ()
{
.....
.....
  if (.....)
  {
      .....
      .....
  }
.....
.....
}
```

Edit View Insert Format Tools Table

12pt ▾ Paragraph ▾ | **B** *I* U A ▾  ▾ T² ▾ |

 ▾  ▾  ▾ |  ▾ |  ▾  ▾  ▾ |   ▾  

Not saved

Submit Quiz