SYS-102 Basic Computer Architecture
Assessment Activity SYS-102:00020
Numeral Systems, Digital Circuits, Interrupts and Polling

Name: (Win) Thanawin Pattanaphol **Student ID**: 01324096

Numeral System and Number Representation

Part 1: Converting Between Numeral Systems

Chosen Number: 5222

a) Convert from **decimal** to **binary**

- Divide the number by 2
- Use the result from the division for the next calculation
- Use the remainder for binary digit
- Repeat the steps above until the result from the division is 0

8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
0	1	0	1	0	0	0	1	1	0	0	1	1	0

Answer: `0b01010001100110`

b) Convert **decimal** to **hexadecimal**

- Convert decimal to binary

    5222 → 0b01010001100110

- Start the conversion from the right-side of the binary number
- Group every 4 digits of the binary number12 pt

    0b01010001100110 → 0b 0001 0100 0110 0110

- Compare each group to the corresponding hexadecimal digits

Binary	Hexadecimal
0b0001	0x1
0b0100	0x4
0b0110	0x6
0b0110	0x6

Answer: 0x1466

c) Convert **decimal** to **octal**

- Convert decimal to binary

    5222 → 0b01010001100110

- Start the conversion from the right-side of the binary number
- Group every 3 digits of the binary number

    0b01010001100110 → 0b 001 010 001 100 110

- Compare each group to the corresponding octal digits

Binary	Octal
0b001	1
0b010	2
0b001	1
0b100	4
0b110	6

Answer: 12146_8

d) Convert the resulting **binary** number (from *a*) to **decimal**

- Fill in the digits into the table seen below

8192	**4096**	**2048**	**1024**	**512**	**256**	**128**	**64**	**32**	**16**	**8**	**4**	**2**	**1**
0	1	0	1	0	0	0	1	1	0	0	1	1	0

- Add up the place values that has the 1 digit in it

    4096 + 1024 + 64 + 32 + 4 + 2 = 5222

Answer: 5222

e) Convert the resulting **binary** number (from *a*) to **octal**

- Group every 3 digits of the binary number

0b01010001100110 → 0b 001 010 001 100 110

- Compare each group to the corresponding octal digits

Binary	Octal
0b001	1
0b010	2
0b001	1
0b100	4
0b110	6

Answer: 12146_8

f) Convert the resulting **binary** number (from *a*) to **hexadecimal**

- Group every 4 digits of the binary number

0b01010001100110 → 0b 0001 0100 0110 0110

- Compare each group to the corresponding hexadecimal digits

Binary	Hexadecimal
0b0001	0x1
0b0100	0x4
0b0110	0x6
0b0110	0x6

Answer: 0x1466

Part 2: Signed Number Representation (Two's Complement)

Previously used number: 5222

a) First two digits: 52

Two's Complement Conversion
- Convert to binary digit first

128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
0	0	1	1	0	1	0	0

- Invert all values (One's complement)

00110100 → 11001011

- Add 1 to previous binary number (Two's complement)

```
11001011
        +
       1
-------------
11001100
=======
```

b) Last two digits: 22

Two's Complement Conversion
- Convert to binary

128	64	32	16	8	4	2	1
0	0	0	1	0	1	1	0

- Invert all values (One's complement)

00010110 → 11101001

- Add one to the digit (Two's complement)

```
11101001
        +
       1
-------------
11101010
========
```

- Two's Complement of 22_{10} is 11101010
- Add to the result of *a*

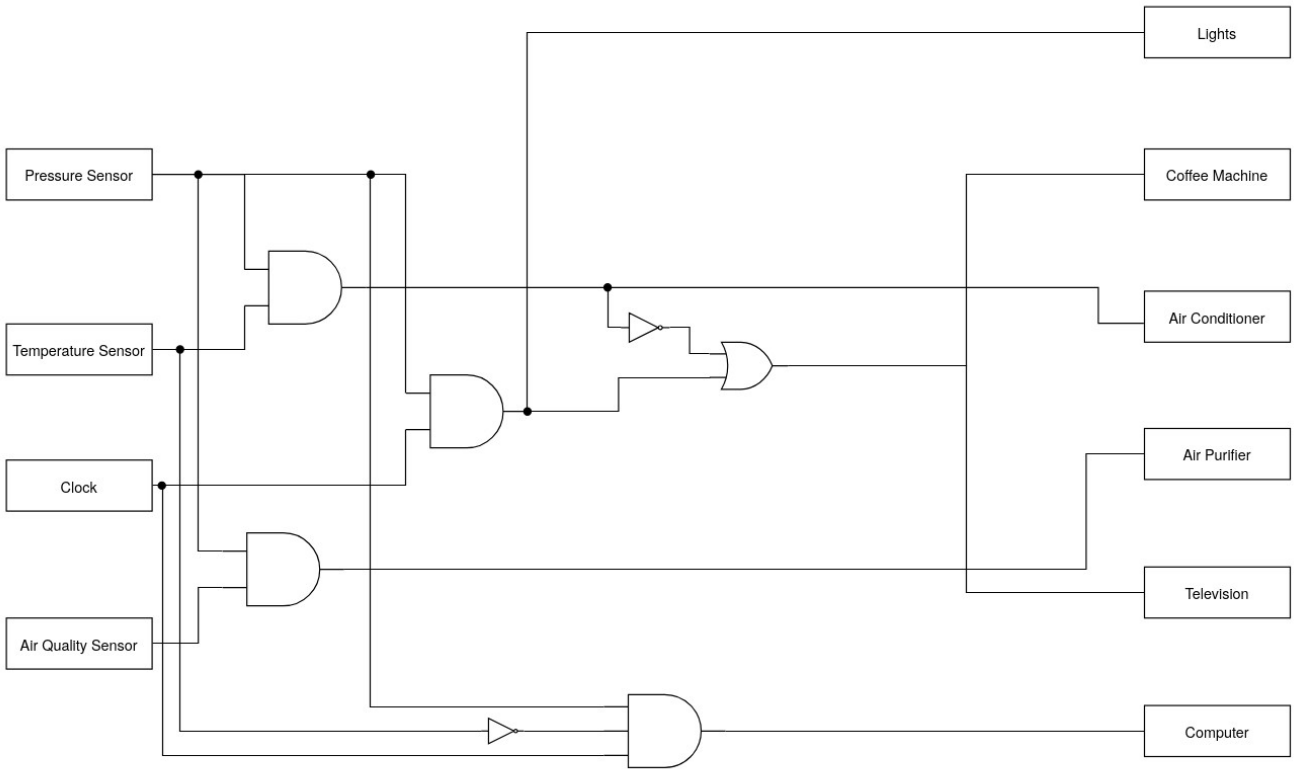
```
11001100
        +
11101010
-------------
110110110
=======
```

This causes as overflow: The sum of the two 8-bit binary digits produce a 9-bit binary digit which cannot be stored in a 8-bit number.

Digital Circuit Problem-Solving Assessment

Problem: When people arrive home, they would have to manually go through the process of turning the lights on, making your own coffee, turning on your air conditioner, and other types of electronics – with this circuit, these processes will be done automatically the moment you step into the house.

Solution:



Inputs:

1. Pressure Sensor: Used for checking if a person had stepped on the front mat of their front door: 1 means pressured applied and 0 means no pressure
2. Temperature Sensor: Used for measuring the temperature outdoors: 1 means temperature is hot and 0 means temperature is not hot
3. Clock: Used for checking a specific time: 1 means it is currently evening and 0 means it is not
4. Air Quality Sensor: Used for checking the outside air quality: 1 means air quality is at unhealthy levels and 0 means air quality is at healthy levels

Outputs: Lights, Coffee Machine, Air Conditioner, Air Purifier, Television and Computer | 1 means on and 0 means off

Interrupt and Polling in C Programming Assessment

Polling

```
#include <stdio.h>
#include <termios.h>
#include <unistd.h>
#include <time.h>

int main()
{
        clock_t start, end;
        double cpu_time_used;

        // Set start to current CPU time
        start = clock();

        int c;
        int flag = 0;
        int laps = 0;

        /*
         THIS IS A REPLACEMENT FOR THE _KBHIT FUNCTION
        */
        // These structs are for saving tty parameters
        static struct termios old_term, new_term;

        // Get parameters associated with the tty
        // Set the new tty parameters to the old one
        tcgetattr(STDIN_FILENO, &old_term);
        new_term = old_term;

        // Set the new tty parameters to canonical mode
        new_term.c_lflag &= ~(ICANON);

        // Set tty attributes
        tcsetattr(STDIN_FILENO, TCSANOW, &new_term);

        // Get character input by user
        while ((c = getchar()) != EOF && laps < 10)
        {
                // Flip the flag
                flag = !flag;
                laps++;

                printf("\n");
                printf("Flag: %d\n", flag);
        }

        // Done retriving input
        // Set tty parameters back to old tty parameters
        tcsetattr(STDIN_FILENO, TCSANOW, &old_term);

        end = clock();
        cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;

        printf("CPU Time: %f\n", cpu_time_used);

        return 0;
}
```

Explanation

This code is designed for polling keyboard input by routinely checking for user input using the getchar() function in standard C library along with termios.h header functions due to the fact that the kbhit function from the conio.h header file does not exist in the standard C library (glibc). Every time the user hits a key on the keyboard, the flag will switch between 0 and 1, the state will be printed and the program ends when the user hits the 10^{th} key.

The program also includes a clock() function that returns the clock ticks that has elapsed since the program was launched. We assign the start and end variable (start for the clock tick when the program starts and end for the click) and subtract them and divide the result by the CLOCKS_PER_SEC variable which produces the CPU time in seconds of the program.

Results

The program was ran repeatedly 10 times and calculated the average time the polling program took to poll 10 key hits. The results are as follows:

Instance	CPU Time (seconds)	Outlier
1	0.000419	Yes
2	0.000380	No
3	0.000372	No
4	0.000393	No
5	0.000374	No
6	0.000371	No
7	0.000341	No
8	0.000420	Yes
9	0.000398	No
10	0.000347	No
Average	0.0003815 (Raw)	0.0002976 (Without Outliers)

Interrupts

```
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <time.h>

// Declare Variables
int flag = 0;
int lap = 0;

// Handling signal function
void handle_signal(int signal)
{
	// If signal is SIGINT
	if (signal == SIGINT)
	{
		// Set flag to its complement
		// Print the flag value
		flag = !flag;
		lap++;
		printf("Flag: %d\n", flag);
	}
}

int main()
{
	clock_t start, end;
	double cpu_time_used;

	start = clock();

	// Handling signal
	signal(SIGINT, handle_signal);

	// Let the program run until interrupt
	while (lap < 10)
	{
		sleep(1);
	}

	end = clock();
	cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;

	printf("CPU Time: %f\n", cpu_time_used);

	return 0;
}
```

Explanation

This program creates an ISR (Interrupt Service Routine) using the signal() function provided by the signal.h header and is used to check for SIGINT signals (Interrupts) which can be sent to the program using the CTRL+C keyboard shortcut. The program checks if the signal passed to the handle_signal function is a SIGINT signal, if so, the program toggles the flag / state and assign to its complement value as well as incrementing the lap variable.

Results

The program was ran repeatedly 10 times and calculated the average time the polling program took to poll 10 key hits. The results are as follows:

Instance	Time Taken (seconds)	Outlier
1	0.000367	No
2	0.000422	Yes
3	0.000391	No
4	0.000448	Yes
5	0.000341	No
6	0.000331	No
7	0.000370	No
8	0.000394	No
9	0.000389	No
10	0.000370	No
Average	0.0003823 (Raw)	0.0002953 (Without Outliers)

Comparison

The time taken to run both programs are very similar where the polling program took 0.0002976 seconds and the interrupt program took 0.0002953 seconds with a difference of 0.0000023 seconds or 2.3 μs.

In summary, polling is mostly recommended to be used in context where it is more simple to implement or when working with devices that do not make frequent interrupts, however, has more latency. Interrupts, on the other hand, is mostly recommended to use in context where efficiency is crucial and where its immediate responsiveness is needed for real-time applications.