EEE3096S 2023 Tutorial 2 Hand-in FRSKIA001 CAMCLR007

Chat GTP was used during the sectorial to aid gathering of answers and writing and meat format.

(openAI, 2023)

1 I2C

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| start | s1 | s2 | s3 | s4 | s5 | s6 | s7 | R/!W | ANK/ NANK | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R/!W | ANK/ NANK | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | ANK/ NANK | stop |
|  | **Slave Address** | | | | | | |  | **Acknowledge Bit** |  | | | | | | |  |  |  | | | | | | | |  |  |

**The I2C message structure when the master communicates with a slave is as follows:**

* **Start Condition: Master initiates communication by sending a start condition (S).**
* **Slave Address + Write Bit: Master sends the 7-bit slave address (8 bits in total) along with the write bit (0).**
* **Acknowledge Bit: The addressed slave responds with an acknowledge bit (ACK/NACK).**
* **If required a registered B is also used to specify the location where the data must be sent to in the slave.**
* **Data: Master sends the data byte to the slave.**
* **Acknowledge Bit: Slave responds with an acknowledge bit.**
* **Stop Condition: Master sends a stop condition (P) to end the communication.**

**2.**

**Advantages of I2C over SPI**

**Two advantages of I2C over SPI are:**

* **Reduced Pin Count: I2C requires fewer pins for communication compared to SPI (only two pins: SDA and SCL), making it suitable for devices with limited pins or space.**
* **Multi-Master Support: I2C supports multiple masters on the same bus, allowing for more flexible and complex communication scenarios compared to SPI, which typically operates in a master-slave configuration.**
* **Ubiquitous i2c is found nearly everywhere and has a lot of support online at with lots of tutorials and other learning options**

**3.**

**Start condition**

**This Acores when the SDA line has a high-to-low transition and SCL line is hight.**

**A high-to-low transition on the SDA line while the SCL is high defines a START condition.**

(Jonathan Valdez, Jared Becker, 2015)

**A close up of a diagram

Description automatically generated**

(nandland, 2015)

**Stop condition**

**A low-to-high transition on the SDA line while the SCL is high defines a STOP condition.** (Jonathan Valdez, Jared Becker, 2015)

****

(htt)

**4.**

# Binary Coded Decimal (BCD)

1. Explain what Binary Coded Decimal is, and provide an example.

Binary Coded Decimal (BCD) is a binary-encoded representation of decimal numbers where each decimal digit is encoded using a fixed number of binary bits. In BCD, each digit is represented by its corresponding 4-bit binary value, ranging from 0000 (decimal 0) to 1001 (decimal 9).

For example, the decimal number 745 in BCD would be represented as:

Hundreds place (7): 0111

Tens place (4): 0100

Units place (5): 0101

So, 745 in BCD is 0111 0100 0101.

1. Advantage

Allows for easy communication between 10th order decimal-based systems and binary systems. It's easier to understand and create outputs for humans.

Disadvantage

The BCD arithmetic is little more complicated. BCD needs more number of bits than binary to represent the decimal number. So BCD is less efficient than binary. (tutorialspoint, 2023)

# Unix Epochs

1. Unix time is referencing computers that count up in seconds using an integer starting from the date of January 1, 1970, at 00:00:00 UTC (Coordinated Universal Time). And this is the way that computers used to keep track of time.

Epoch time is used in computing to facilitate date and time calculations, comparisons, and storage, as it provides a consistent and simple way to represent time as a numerical value. It is especially useful for handling time-related operations in programming, file timestamping, and other applications.

1. 1672531200 (converter, n.d.)

# RTS and RISC/CISC

1.

Dynamic Synchronous Hard Real-time System

A "Dynamic Synchronous Hard Real-time System" refers to a computing system that must perform tasks within strict timing constraints while dealing with dynamic and changing situations. Here's a breakdown of the terms:

Dynamic: The system adapts and reacts to changing conditions or inputs.

Synchronous: The system follows a predefined clock cycle and timing to ensure predictable behavior.

Hard Real-time: The system needs to guarantee that certain tasks are completed within specific timing constraints.

In other words, this type of system is designed to handle real-time tasks in an environment where timing and responsiveness are critical, even when conditions change dynamically.

2. Differences Between RISC and CISC Architectures (Instruction Set Size and Memory Access for Operands)

Instruction Set Size:

RISC (Reduced Instruction Set Computer) and CISC (Complex Instruction Set Computer) architectures have different approaches to instruction set design.

RISC: RISC architectures have a smaller and simpler instruction set with a focus on executing instructions quickly. Each instruction performs a simple operation, and complex operations are often performed using combinations of simpler instructions. This leads to a reduced instruction set size.

CISC: CISC architectures have a larger and more varied instruction set that includes complex and specialized instructions. CISC instructions can perform multiple operations in a single instruction, which can lead to larger instruction sizes to accommodate the complexity.

Memory Access for Operands:

RISC and CISC architectures also differ in how they handle memory access for operands.

RISC: RISC architectures tend to follow a load/store architecture, where most operations are performed using registers, and data is explicitly loaded from memory to registers before processing and stored back after processing. This approach simplifies instruction decoding and pipeline stages.

CISC: CISC architectures often allow memory-to-memory operations, where instructions can directly operate on data stored in memory locations. This can lead to more complex instruction decoding and memory access patterns, potentially impacting the pipeline efficiency.

In summary, RISC architectures prioritize simplicity and efficiency with a smaller instruction set and load/store approach to memory access, while CISC architectures aim to provide more functionality within each instruction, potentially leading to larger instruction sets and more flexible memory access modes.

From notes

RISC VS CISC

• RISC: Reduced Instruction Set Computer

• CISC: Complex Instruction Set Computer

• RISC has fewer instructions, therefore needs more code/software

• CISC has more instructions, therefore needs less code/software

• Majority of today’s microprocessors are RISCs

CHARACTERISTICS OF CISC

• Large number of instructions (100 – 250 instructions)

• Some instructions perform specialized tasks and are used only infrequently

• Many different addressing modes (say 5 or more)

• Variable-length instruction formats

• Possibly variable length instruction execution cycles (e.g., ADD taking

more clock cycles than LOAD)

• Instructions that manipulate operands directly in memory (e.g. add value

at address X to reg. A)

CHARACTERISCTICS OF RISC

l Small number of instructions (usually <100)

l Just enough types of instructions (or close to minimal)

l Few addressing modes (maybe be just two or three)

l Fixed-length instruction formats, easy to decode

l All instructions take the same number of clock cycles to complete (typically)

l All operations (e.g., ADD, COMPARE, etc) work only with registers (no memory

access)

l Often use micro-programmed control (i.e., CPU instructions a set of even lower level

instructions)

l Memory access limited to LOAD and STORE instructions (i.e. no instructions

manipulate operands in memory