Computer Science - Core Skills

Semester 1

BSC109224 Group A

Assignment 2

Student: Lucas Madeira Maranho

Student ID: 76990

Lecturer: Bernard Joseph Roche

Question 1 (25%): Number systems

Binary to decimal:

● Convert your student number to a binary number and write it out.

Student number:76990(Decimal)
Student number: 10010110010111110(Binary)

76900/2 = 38495 remainder is (0)	150/2 = 75 remainder is (0)
38495/2 = 19247 remainder is (1)	75/2 = 37 remainder is (1)
19247/2 = 9623 remainder is (1)	37/2 = 18 remainder is (1)
9623/2 = 4811 remainder is (1)	18/2 = 9 remainder is (0)
4811/2 = 2405 remainder is (1)	9/2 = 4 remainder is (1)
2405/2 = 1202 remainder is (1)	4/2 = 2 remainder is (0)
1202/2 = 601 remainder is (0)	2/2 = 1 remainder is (0)
601/2 = 300 remainder is (1)	1/2 = 0 remainder is (1)
300/2 = 150 remainder is (0)	0/2 = 1 remainder is (0)

Converting decimal numbers to binary, you must divide the number by 2. When the division is made and the final number is integer, the remainder number must be 0, but if the division gets a not integer number, you round the number down and the remainder number must be 1. You must divide the number until gets 0 in the division results.
When the converting is finished, the result must be done backwords, starting with the last to the first one.

● Convert the following decimal numbers to 8-bit binary:

Decimal	Binary	Decimal	Binary
0	00000000	256	100000000*
1	00000001	128	10000000
2	00000010	64	01000000
3	00000011	32	00100000
4	00000100	16	00010000
5	00000101	8	00001000
6	00000110	4	00000100
7	00000111	2	00000010
8	00001000	1	00000001

*In the decimal 256 the binary number contains 9-bit instead of 8-bit, so I added one more bit for it.

Question 2 (25%): GIT

- As a lecturer to a group of first-year Computer Science students, write a clear and concise paragraph in your own words explaining the concept of version control, the Git software tool, and its practical implications.
- Use analogies and examples to help clarify your explanation and make it relatable to the students. Ensure that the examples you provide are original and not drawn from those discussed in class or previous assignments.

Answer: (455 words)

Git is a version control system created by Linus Torvalds in 2005.
Through them we can develop projects in which people can work in the same project simultaneously, editing and creating new files and allowing them to exist without causing any risk of their changes being overwritten.

If the version control did not exist, imagine the chaos between two people opening the same files at the same time. One of the good points about GIT is allowing the archive being opened for different people at the same time.
It looks complicated but it keeps everything organized to make the developers life easier avoiding problems.
Another positive point about GIT is the possibility of creating, at any time a snapshot(branch) of your project. For example:

Let's suppose that the project you are creating is a HTML site, and you want to create a new session in your HTML code, but at that specific time you don't want these changes being available to anyone else but you. You want to change the project but not turn it official yet for other people, so you create a branch as a copy and just work in this branch until you get all the details right. After that you can merge the branch back into the official project.

GIT COMMANDS:

- **Commands you will always use from now on**:

1. Git add <files…>
   This command adds the files on a place called INDEX.
   When you add the files, it doesn’t mean that you are adding the file in a new repository, but you mean the file is being prepared to get into the next revision of the repository.
2. git commit -m “any commit”
   this command commits all the files that are in that INDEX place that the command to add, add and create a revision with comments where everyone can read.
3. git push

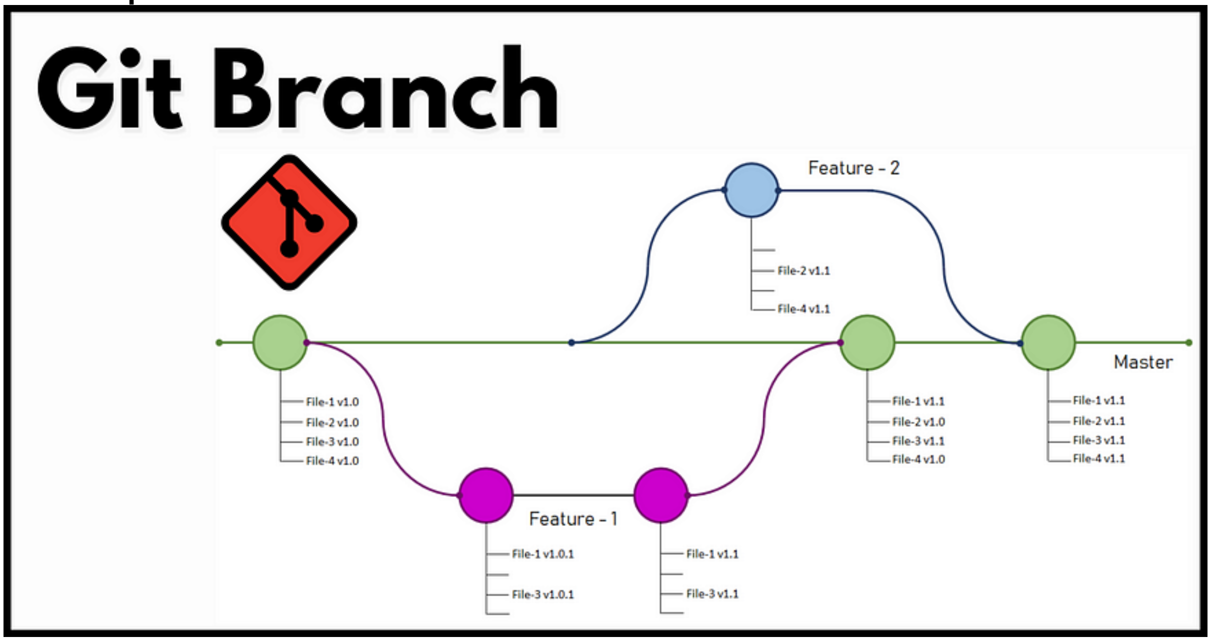
it shares all the commits to GitHub.

4. git status
   it shows the actual repository of your status.

BRANCHES:

Branches and mergers used to be a very complicated thing to the system, but after git, branch turned to something very simple and easy to manage.
Creating a branch:
Imagine that you have a site ready, everything is working perfectly, but then you decide to change some parts of it, to make the website better. Although, you need to keep these changes on your home and work computer. We might have a problem, if you start changing your files from home and need to finish at work you couldn't commit and let the website incomplete.
That's why branch exist, to separate your projects in 2, as each one were a repository and then put together later.

Example of Git branch:



Git branch - https://medium.com/@meghasharmaa704/-25c59f32fb91

Question 3 (25%): GitHub

- You are a lecturer teaching a group of new Computer Science students. Your task is to write, in your own words, an explanation of GitHub. You may use GitLab or Bitbucket as alternatives if you prefer.

- To make your explanation more relatable, use analogies and examples. Please avoid repeating examples provided by your teacher in class or used in previous answers.

Answer: (449 words)

GitHub is a platform that stores git files in the cloud that make sharing and collaborating on open codes projects easier and more accessible.
Developers can contribute to projects that already exist or collaborate on new projects and discover tools or codes that can help them with their own projects.
Additionally, GitHub offers features like issue management, pull request and constantly helping developer team work much more efficiently and effective.

GitHub was created by four developers: Chris Wanstrath, PJ Hyett, Tom Preston-Werner and Scott Chacon and Officially launched in 2008.

GitHub allows developers to change, adapt and improve software from its public repositories for free, but there are various paid plans.
Private and public repositories contain all project's files and file's revision history.
Repositories can have lots of collaborators and owners.
Members can follow each other, evaluate other's work. They can communicate publicly or privately.

It's not just a place to host repositories, the platform maintains the of all activities done by users, like commits, reports and help to other repositories.

There are different contexts to use GitHub to collaborate on software projects:
Businesses: Use GitHub as a version control system. This lets different co-workers developing on a project all the same time and guarantee that they are working on the latest version of the code.
GitHub allows code sharing between developers because code is stored in a central location.

Programming students: Students use the platform to learn web development, work on development projects as practice and host virtual events.

Nonprogrammers: They use the platform GitHub to work on documents and multimedia projects. The platform and the version control tools are useful for collaboration.

Benefits of GitHub

It’s a way that developers can work together in a centralized Git repository and verify the changes going forward to stay organized.
It offers more security and auditability.

Other products and features of note include:

GitHub Pages – webpages to host projects, pulling information from an individual’s or organization’s GitHub repository.

GitHub student developer Pack – Free developer tools offered for students.

GitHub Gist – where users can share some codes or notes.

GitHub Codespaces – Cloud-based development environment that users access to common programming languages and tools.

There are some common terms teams will need to understand when using GitHub.

Git - Tool that allows developers to use version control.

Repository – A folder where all files and their version histories are stored.

Commit changes – Saved record of a change made to a file within the repo.

Pull Request – Request for changes made to a branch to be pulled into another branch.

Merge – after a pull request is approved, the commit will be merged from one branch to another on the live site.

Question 4 (25%):

Explain the following terms Instructions: For each of the following terms, provide a detailed explanation in your own words:
1. FDE (Fetch Decode Execute)
2. BDI (Belief Desire Intention)
3. The C programming language
4. POST (Power On Self Test)
5. Machine code vs assembly
Use analogies and examples to make your explanations clearer and more relatable. Avoid using examples provided by the teacher in class or used in earlier answers.

Answer: (1077 words)

1. FDE (FETCH DECODE EXECUTE):

It's a process that the CPU does repeatedly to process instructions.
a) FETCHING: Instructions from memory - Sending the address and receiving instructions from memory.
b) DECODING: the instructions – interpreting the instructions, reading and taking back the instructions the required data from their address.
c) EXECUTING: The CPU performs the required action.

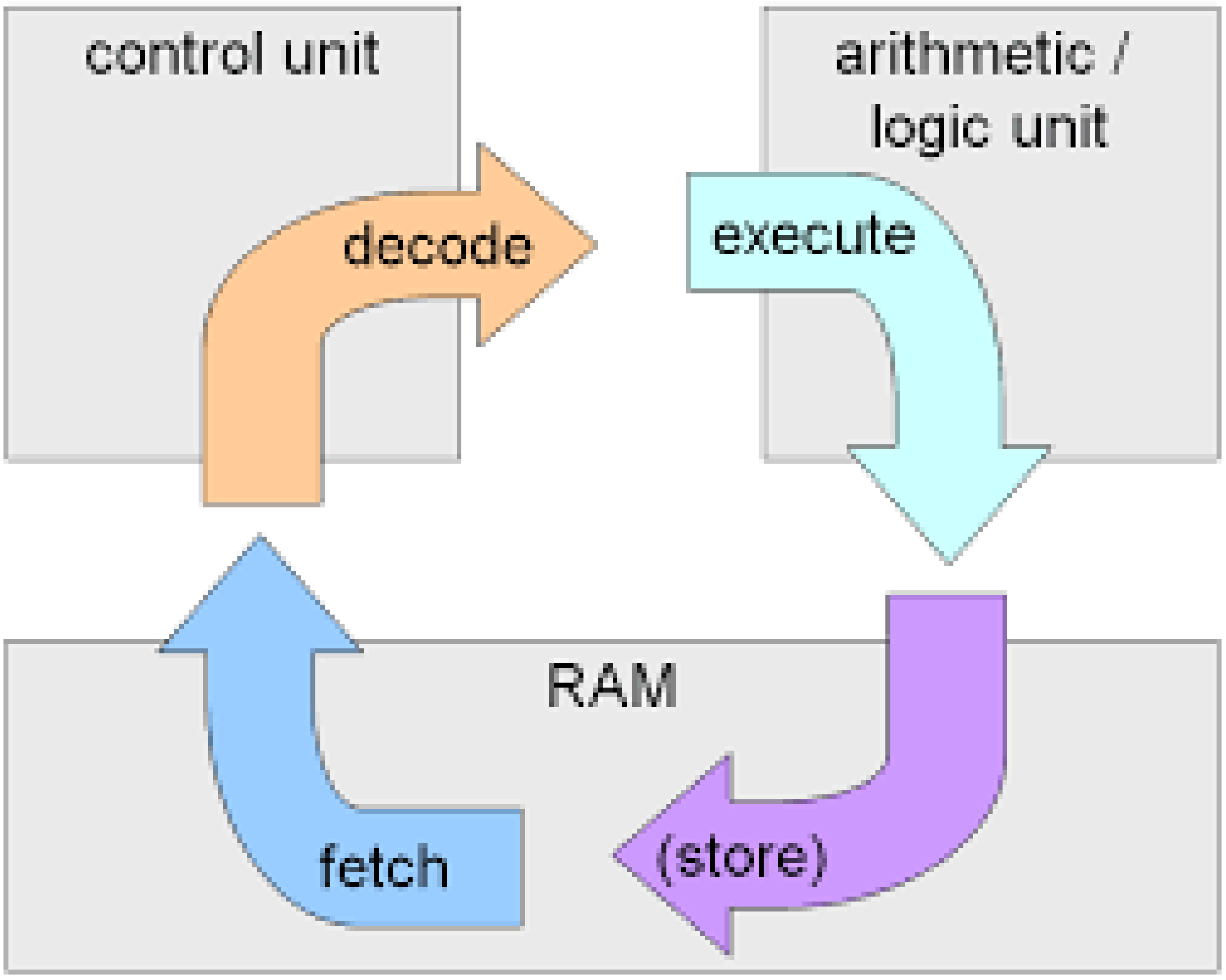
During the Fetch-Decode-Execute cycle, the following steps happen:

- **FETCH**:
  - The computer is loaded with 0
  - The value (0) is copied to the MAR (Memory address register)
  - The data from the MAR (0) is sent across the address bus with the instructions to read the data sent across the control bus.
  - The data from that location in memory is sent down the data bus to the MDR (Memory Data Register).
  - The computer is added by 1.

- **DECODE:**
  - The data is sent from the MDR to the CIR (Current Instructions Register) where is separated into the opcode and operand.
  - It's sent to the CU (Control Unit) to be decoded.

- **EXECUTE:**
  The instructions can now be executed. Arithmetic and logical instructions are carried out using the accumulators in a CPU.
  Signals are sent to different parts of the CPU to execute the instructions.
  Registers will depend on the instruction being executed.
  - If a value is added to or subtracted from another value (ADD/SUB)
  - If a value is being inputted (INP) the ACC will store the value.

- If the value is to be stored (STA) will take the value from the ACC, send it to the MDR and then send it across the data bus to RAM(to the address location in the MAR).



Fetch decode execute cycle - https://www.testandtrack.io/usa/index.php/studenttest/givetest/1267

2. BDI (BELIEF DESIRE INTENTION):

BDI agent model is a framework for developing intelligent agents that simulate human-like reasoning and decision-making processes.

It is based on Michael Bratman's philosophical theory of practical reasoning, which involves 3 attitudes: beliefs, desires and intentions.

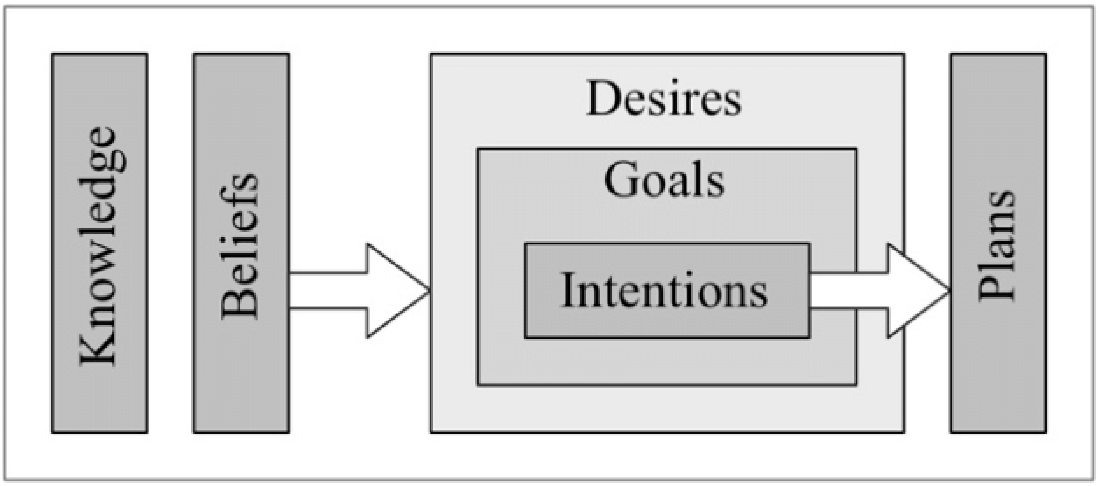
a) **BELIEF**: These represents the understanding of the world, including itself. They are the facts or knowledge. What it believes to be true. Beliefs can be updated as the agent realize changes in the environment or receive new information.
b) **DESIRES**: These represent goals, preferences and values, motivation of the agent. They represent the ideal state of the environment.

An agent can have lots of desires, which may sometimes be in conflict with one another. Desires are taken decisions making a target state about the agent aims to reach.

c) **INTENTIONS**: These represents the plans, strategies and actions. They are the commitments to desires.

The BDI model uses process to reason about these beliefs, desires and intentions.
The process involves 3 main steps: belief revision, goal generation and plan selection.

In the first step belief revision, updates beliefs based on new information or changes in the environment.
In the second step goal generation, identifies the goals and generates new goals based on desires.
In the third step plan selection, chooses a plan to achieve the goals based on intentions.
BDI also has mechanisms to handle with unexpected events or failures.
If plan fails, the agent can revise its beliefs and intentions and make new plans to achieve the goals.



BDI - https://www.researchgate.net/figure/Belief-Desire-Intention-model-for-reasoning-procedure_fig1_313617511

3. THE C PROGRAMMING LANGUAGE:

The C programming language can be considered middle level because it supports the feature of both low- and high-level languages. C language program is converted into assembly code, it supports low level, but it is a machine independent (a feature high level).
A program written in C must be run through a C compiler to convert it into an executable that a computer can run.
Nowadays, the C programming language runs on many different hardware platforms OSes such as Microsoft and Linux.

- Pros of the C language:
  - Structured – it allows you to break down problems into smaller parts or functions that are easy to understand and modify.
  - Portable – It's machine independent, it means that C programs can be executed on different machines.
  - Extensible – it can be easily extended. If the code is already written, new features and functionalities can be added to it.

- Cons of the C language:
  - Garbage collection – C is not equipped with garbage collection.
  - Constructor and destructor – C is not object oriented so constructing or destructing a variable in C must be done manually through a function by other means.

4. POST (POWER ON SELF TEST):

Performs tests on the main components, such as the CPU and memory. POST check the low-level interaction between the CPU, caches, memory, JBus and the |/O bridge chip.
It is a set of routines executed by firmware or software immediately after a computer is powered on, to determine if the hardware is working as expected.
The POST sequence runs independently of the operating system and is controlled by the system BIOS. When the tests passed the POST notify the OS with beeps, the number of beeps can vary from system to system. When POST is successfully finalized, bootstrapping is enable, so starts the initialization of the OS.

THE ROLE OF POST IN THE BOOT SEQUENCE:

The boot sequence is the process of starting a computer/system.it is initiated when the button is pressed.it sends power to the boot-loader in the cache memory. If everything is working well in the BIOS (Basic Input output system) is activated.
The software interacts with the hardware units to complete the process. To avoid hardware errors while a software is executing a program, the pre-boot sequence would test the hardware and initiate the OS, if the basic hardware units are working as expected. The main functions of the main BIOS during POST are:

1. Find, size and verify the system main memory.
2. Initialize BIOS.
3. Identify, organize and select devices available for booting.
4. Verify CPU registers.
5. Integrity of the BIOS code itself.

5. MACHINE CODE VS ASSEMBLY:

Machine code: Is the binary code represented by 0s and 1s that computers directly understand and execute.
The 1s represents the true or on states while the 0s represents the false or off states.
For example: to represent the number 40, the binary value 101000 is used.

Assembly: It uses symbols and abbreviations instead of binary numbers. For example:
Add to addition, Sub for subtraction, Mul for multiplication.
It is more human-readable, but it is not understandable by CPU directly, it is necessary a translator to convert mnemonic codes into machine language.
Both are low-level programming languages which machine code being the lowest one.