|  |
| --- |
| http://www.utar.edu.my/mmlm/img/TARC_UC_small.jpg |
| **Faculty of Computing and Information Technology** | |
| **Assignment**  **OCT 2019 Semester** | |
| |  |  |  | | --- | --- | --- | | **Course code** | **:** | AACS3064 | | **Course Title** | **:** | Computer Systems Architecture | |  |  |  | | |
| |  |  |  |  | | --- | --- | --- | --- | | **Students’ Name &**  **ID No.** | **:** | Name: Lim Jun Rong | ID. No.: 18WMD04952 | | Name: Ong Shannen | ID. No.: 18WMD05273 | | Name: Hiew Long Shun | ID. No.: 18WMD06275 | | **Programme\*** | **:** | DCO2 | | | **Tutorial Group** | **:** | 5 | | | **Tutor** | **:** | Mr. Loh Kiean Nyak | | | **Submission Date** | **:** | **Week 12, Friday, before 12 noon** | | |  | | | | | |
| |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **Members’**  **Name** | **Introduction** | **Coding & logic** | | | **I/O design** | **Leadership and teamwork** | **Total** | | | **( 15marks)** | **(40marks)** | | | | **(5 marks)** | **(60 marks)** | | |  |  |  | | |  |  |  | | |  |  |  | | |  |  |  | | |  |  |  | | |  |  |  | | | **Comment:** | | | | | | | | | **Date of submission** | | | **:** |  | | | | | **Date received**  **(to-be filled by the tutor received)** | | | **:** |  | | | | | |

Semester: 2 Course Code & Title: AACS3064 Computer Systems Architecture

**Declaration**

**I/We confirm that I/we have read and shall comply with all the terms and condition of TAR University College’s plagiarism policy.**

**I/We declare that this assignment is free from all forms of plagiarism and for all intents and purposes is my/our own properly derived work.**

**I/We further confirm that the same work, where appropriate, has been verified by anti-plagiarism software DupliChecker *(please insert).***

Signature(s):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [By all members]

Name(s): Lim Jun Rong, Ong Shannen, Hiew Long Shun [By all members]

Date: 3-January-2020

**Table of Contents**

[Part 1 Introduction about the system 3](#_Toc29137723)

[*1.0 Industry Selected* 3](#_Toc29137724)

[*1.1 Company Background* 3](#_Toc29137725)

[1.1.1 Introduction 3](#_Toc29137726)

[1.1.2 Products 3](#_Toc29137727)

[1.1.3 Company Outlook 3](#_Toc29137728)

[1.1.4 Challenges 3](#_Toc29137729)

[*1.2 Proposed Function* 4](#_Toc29137730)

[1.2.1 Login 4](#_Toc29137731)

[1.2.2 Purchase function 4](#_Toc29137732)

[1.2.3 Billing function 4](#_Toc29137733)

[*1.3 Proposed Formula* 4](#_Toc29137734)

[1.3.1 Login 4](#_Toc29137735)

[1.3.2 Purchase function 4](#_Toc29137736)

[1.3.3 Billing function 4](#_Toc29137737)

[*1.4 Assumptions* 5](#_Toc29137738)

[1.4.1 Login 5](#_Toc29137739)

[1.4.2 Purchase function 5](#_Toc29137740)

[1.4.3 Billing function 5](#_Toc29137741)

[*1.5 Flowchart* 6](#_Toc29137742)

[1.5.1 Overall program 6](#_Toc29137743)

[1.5.2 Purchase Module 7](#_Toc29137744)

[1.5.3 Billing Module 8](#_Toc29137745)

[Part 2 Coding & logic 9](#_Toc29137746)

[*2.1 Login Encryption Feature* 9](#_Toc29137747)

[2.1.1 Operation Example 9](#_Toc29137748)

[2.1.2 Algorithm 10](#_Toc29137749)

[2.1.3 Code snippet 10](#_Toc29137750)

[*2.2 Four-digit number display* 11](#_Toc29137751)

[2.2.1 Operation Example 11](#_Toc29137752)

[2.2.2 Algorithm 12](#_Toc29137753)

[2.2.3 Code Snippet 12](#_Toc29137754)

[*2.3 Five-digit number calculation* 15](#_Toc29137755)

[2.3.1 Operation Example 15](#_Toc29137756)

[2.3.2 Algorithm 17](#_Toc29137757)

[2.3.3 Code Snippet 18](#_Toc29137758)

[Part 3 I/O Design 25](#_Toc29137759)

[*3.1 Login System* 25](#_Toc29137760)

[3.1.1 Sample Data 25](#_Toc29137761)

[3.1.2 Screenshots & Program Execution 25](#_Toc29137762)

[*3.2 Main Menu System* 26](#_Toc29137763)

[3.2.1 Sample Data 26](#_Toc29137764)

[3.2.2 Screenshots & Program Execution 26](#_Toc29137765)

[*3.3 Summary System* 27](#_Toc29137766)

[3.3.1 Sample Data 27](#_Toc29137767)

[3.3.2 Screenshots & Program Execution 28](#_Toc29137768)

[*3.4 Purchase System* 29](#_Toc29137769)

[3.4.1 Sample Data 29](#_Toc29137770)

[3.4.2 Screenshots & Program Execution 30](#_Toc29137771)

[*3.5 Billing System* 35](#_Toc29137772)

[3.5.1 Sample Data 35](#_Toc29137773)

[3.5.2 Screenshots & Program Execution 36](#_Toc29137774)

[Part 4 Leadership and Teamwork Software used 39](#_Toc29137775)

[*4.1 Git VCS* 39](#_Toc29137776)

[*4.2 WhatsApp* 39](#_Toc29137777)

[Part 5 References 40](#_Toc29137778)

[Part 6 Appendices 40](#_Toc29137779)

[Appendix 6.1: How XOR cipher works by The Tech-FAQ 40](#_Toc29137780)

# Part 1 Introduction about the system

## *1.0 Industry Selected*

Agriculture - Meat Producer Company

## *1.1 Company Background*

### 1.1.1 Introduction

HOL Meat Company is a pork meat producer company headquartered and incorporated in Malaysia. It is one of the new participants in the market as a producer and seller of pork meat. HOL Meat Company employs 200 workers to handle its business processes such as slaughtering, cutting, and further processing.

### 1.1.2 Products

HOL Meat Company is one of the more prominent pork meat manufacturers in Malaysia. The company’s main revenue source comes from selling processed pork meat to other distributors. Besides that, HOL Meat Company also offer direct sales to individual customers through its retail stores.

### 1.1.3 Company Outlook

As of 2018, HOL Meat Company grossed an annual income of MYR 300,000, netting a profit of RM100,000. HOL Meat Company company enjoys a high yearly growth due to the increased demand of pork meat, especially from the chinese restaurants where it is served as an exquisite dish. Besides that, by introducing good quality feed, and providing good amount of space for the pork to move around, HOL Meat Company has received positive feedbacks from its clients regarding the meat quality. HOL Meat Company, either directly, or indirectly, supplies pork meat for more than 40% of the restaurants and households in Malaysia.

### 1.1.4 Challenges

As of 2019, HOL Meat Company is still evaluating the weight and price of the pork meat using guesswork and weighing scale. This creates a problem in high volume sales has little to none surge capacity, which is especially limiting during holiday seasons. Besides that, as the company’s popularity increases, the number of staff required also increases linearly. Therefore, the company is now looking for a solution to improve profits and efficiency in handling the evaluation part of pork sales.

## 

## *1.2 Proposed Function*

### 1.2.1 Login

One of the functions of the program is to allow the user to login/logout from the system. This system is in place to prevent unauthorized use of the system. Without this function in place, malicious customers can easily tamper with the operations of the weighing and calculating   process, which can lead to a this system provides a degree of protection to serve as a deterrent for malicious acts.

### 1.2.2 Purchase function

One of the functions of the program is to allow the customer to purchase the meat. Firstly, the user will be prompted to select the desired type of meat (i.e pork ribs, pork loin, pork belly, etc). Then, the user is required to enter the weight (per packet) and quantity. The total weight of meat will immediately be calculated and displayed. After that, the program will ask the user whether he/she would want to edit the weight or quantity. If so, the new total weight of meat is recalculated and shown.

### 1.2.3 Billing function

The billing function is implemented to allow the user to make payment after purchasing and calculating the total weight of meat. In this module, the total price of meat is calculated by multiplying the total weight of meat and the price of meat (per Kilogram) that will be stored in an array. The program will multiply the total weight with the price (per kg) in respect of the type of meat selected. This is because different meat type will have different prices. Next, the user will need to choose to make payment in terms of RM, USD or SGD. The total payment will immediately be calculated and the user is required to enter the amount of payment. Lastly, the balance will be displayed if the user makes the payment successfully.

## *1.3 Proposed Formula*

### 1.3.1 Login

- Password (default): P@ssw0rd

- Key (default): “e”

- Each character stored as hash. Hash = character XOR key

- To match the password, the program is required to compute the XOR hash of user input.

### 1.3.2 Purchase function

- Total weight of meat = Quantity of order \* Weight of meat (per packet)

### 1.3.3 Billing function

- Price = Total weight of meat \* Price of meat (per Kilogram)

- Conversion of Malaysia Ringgit (RM) to United States Dollar (USD) =  Price / 4

- Conversion of Malaysia Ringgit (RM) to Singapore Dollar (SGD) =  Price / 3

## *1.4 Assumptions*

### 1.4.1 Login

- Password is hardcoded into the system, as the system is non-persistent.

- Users need to compute hash of the password themselves (using calculators/online tools) and store   
 into the system.

- Password can only consist of alphabets, symbols, and numbers.

- Password maximum length is limited to 20 characters.

### 1.4.2 Purchase function

- The weight of each packet of meat is set to 1kg, 3kg and 5kg.

- The quantity of order must be a whole number.

- Each customer is limited to a maximum of 999 packets of meat

- Each customer can only purchase one kind of meat when entering the purchase function.

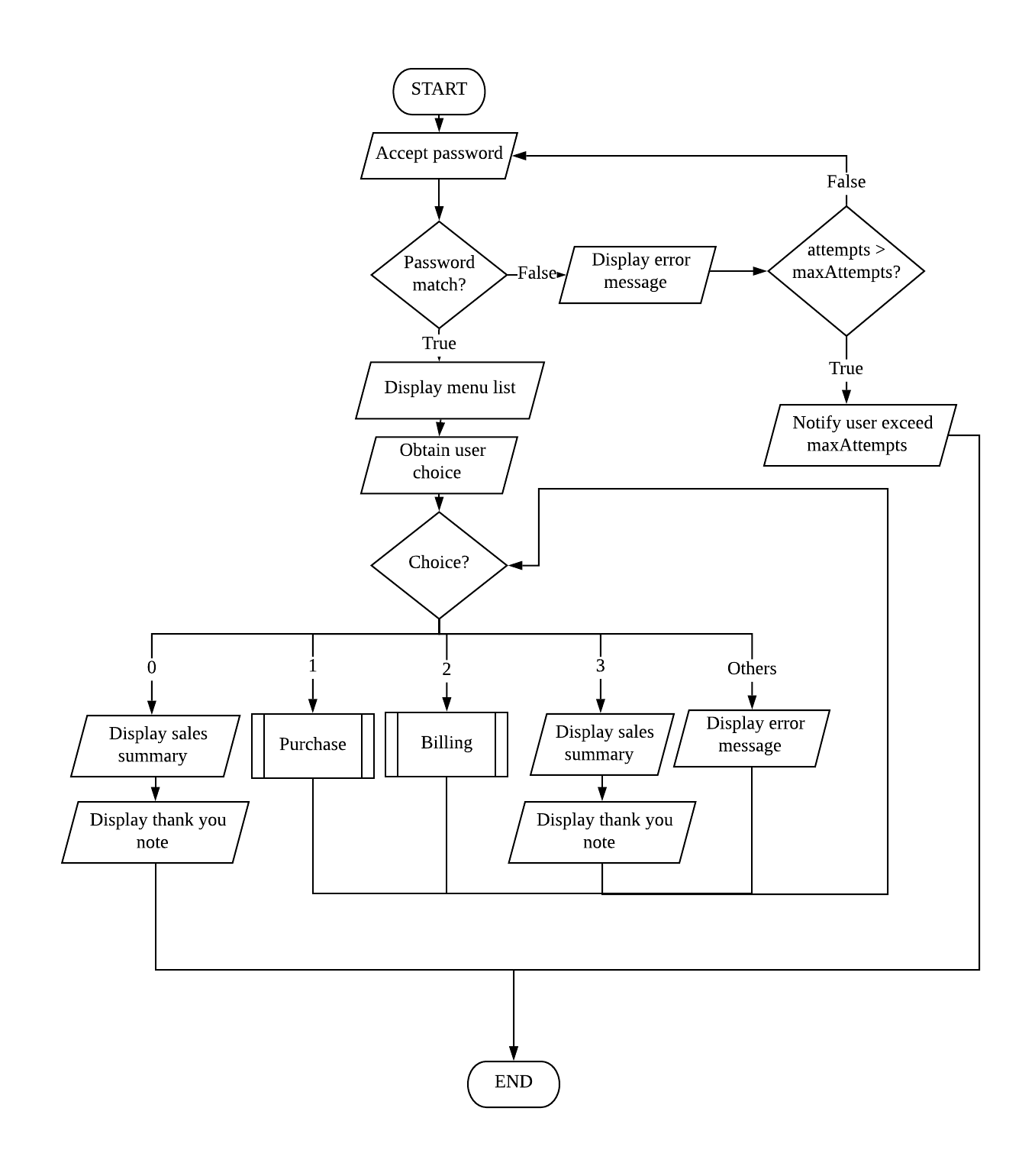
### 1.4.3 Billing function

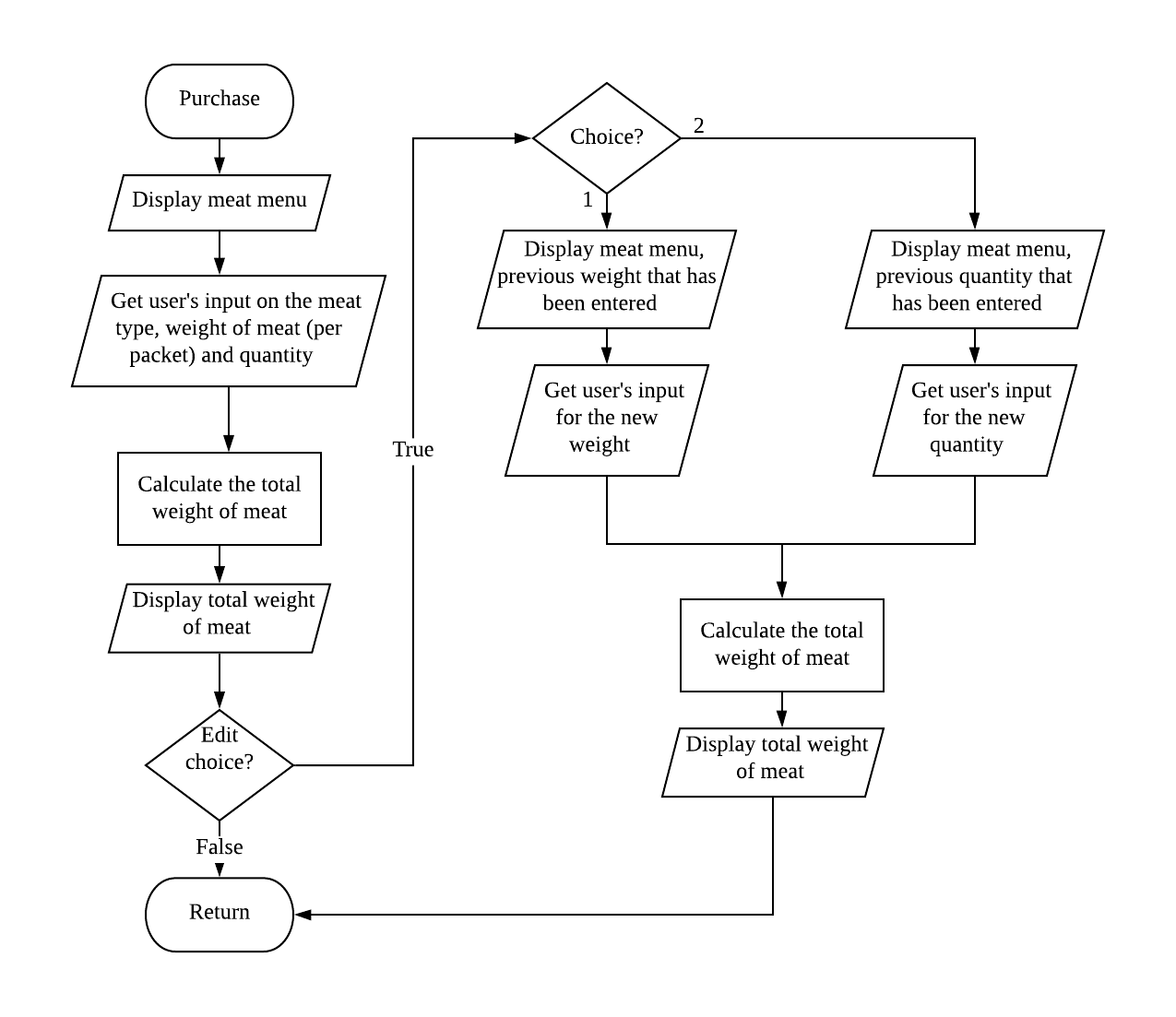
- Assume that the currency exchange rate from Malaysia Ringgit (RM) to United States Dollar (USD) is 4:1(RM 4 ≈ 1 US Dollar).

- Assume that the currency exchange rate from Malaysia Ringgit (RM) to Singapore Dollar (SGD) is 3:1 (RM 3 ≈ 1 SGD).

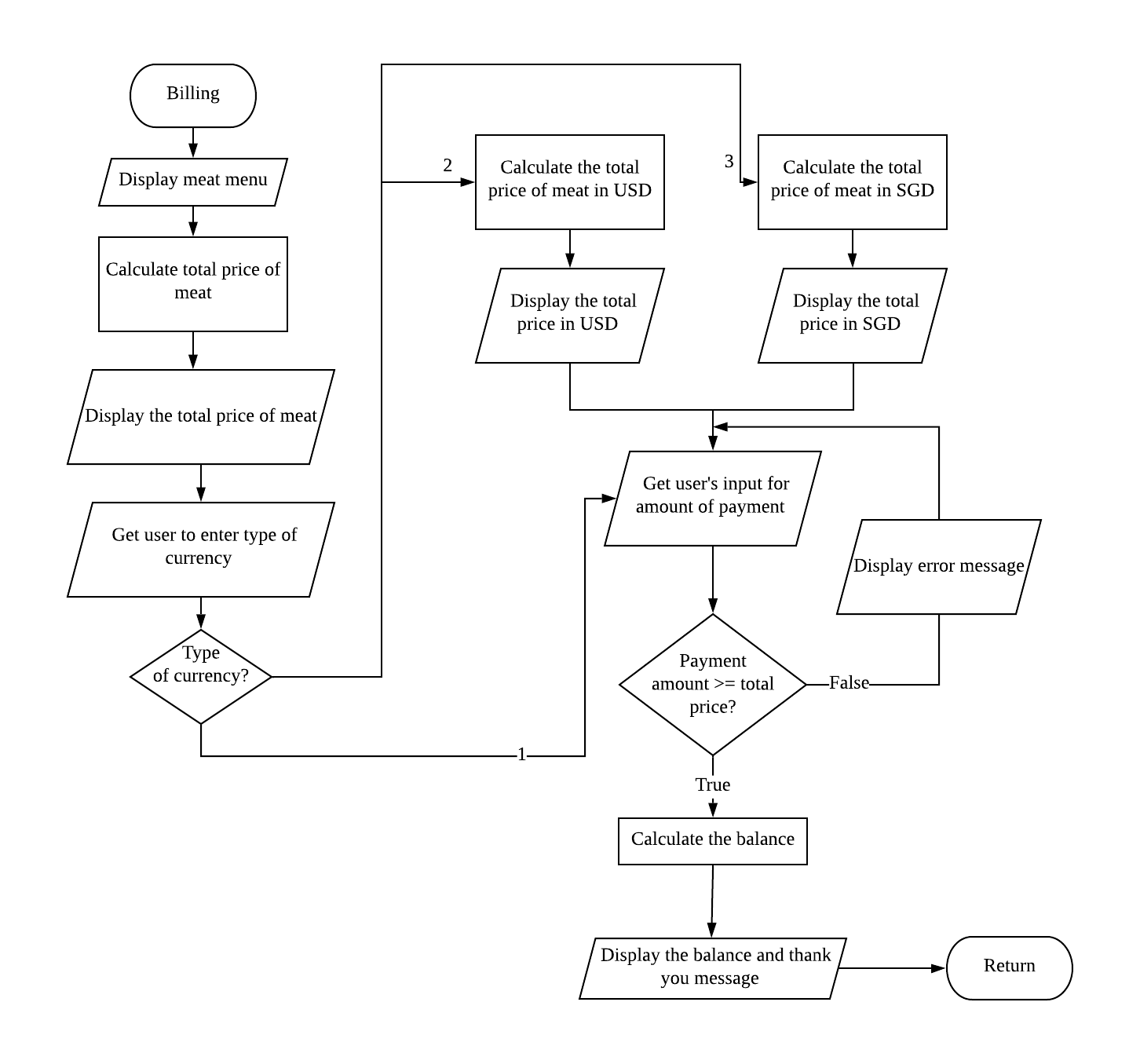
## *1.5 Flowchart*

### 1.5.1 Overall program

****

1.5.2 Purchase Module

### 1.5.3 Billing Module



# Part 2 Coding & logic

## *2.1 Login Encryption Feature*

One of the advanced features from the program is the login encryption & decryption feature. The main purpose of this feature is to prevent unauthorized users from accessing the system.

This feature encrypt the password by using a the XOR fucntion, and stores the password hash along with the key. This process makes the login system difficult to be bypassed through brute-force attacks.

When the user enters a password, each of their characters will have their password XOR-ed (eXclusive OR-ed) with the key for the hash. The output from the function is then compared with the character at the same location inside the password hash.

Besides that, this feature makes it harder for reverse-engineers to peek into the code and retrieve the plain-text password, as the actual password is not stored directly into the system. By doing so, the system’s security is significantly increased.

### 2.1.1 Operation Example

When the password intended to be stored is **pass**, the ASCII representation of the password will be **112 97 115 115** in decimals. This translates to the following binary figures:

01110000 01100001 01110011 01110011

Instead of storing the actual binary representation, this feature will use a key provided to encrypt each binary figure by applying the XOR logical operation (refer to Appendix 1 to understand how XOR works given a set of inputs) using the key. As an example, **e**, in binary, is

01100101

When encrypted, the character will undergo the following calculation, and be stored as **21** in decimal inside the system:

           01110000 ← Input, Binary representation for **p**

XOR  01100101 **←** Key**,** Binary representation for **e**

--------------

**00010101 ← Output,** In ASCII, this will be the **‘NAK’** special symbol, even if this cannot

          be displayed on the screen, it still has a numerical value of **21** which will be

          stored inside the system for comparison.

Therefore, the encrypted password will be stored as:

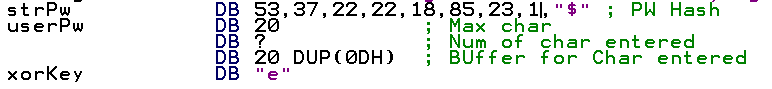
01110001 01110001 01110010 01110010

When the user attempts to logininto the system, their input will undergo the same transformation with the key and be compared with the XORed password stored inside the system. Because XOR is a one-way function, it is very difficult to obtain the actual password from the password hash, thus hardening the system against unauthorized access.

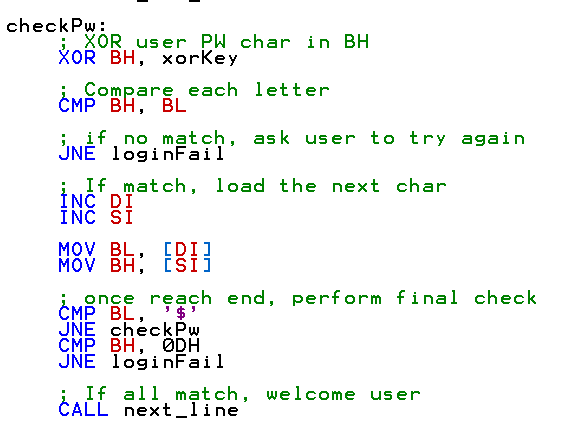
### 2.1.2 Algorithm

1. Obtain a character from user input inside `userPw`
2. XOR the character
3. Compare it with the character in the same location inside `strPw`
4. If matches,
   1. Check if the characters at the same location inside both `strPw` and `userPw` is “$”
      1. If yes, the password is valid, the user is authenticated
      2. If not, the password is invalid, the user is denied access
   2. If not, compare the next character inside both `userPw` and `strPw`

### 2.1.3 Code snippet



1. **strPw:** Stores the password hash of the actual password after being XOR-ed. The password is stored in decimals as some ASCII text cannot be displayed on screen.
2. **userPw:** A buffer for user string. Due to how assembly’s string input (10H) does not allow for input over the maximum characters specified in the first field, no “overflow-checking” is required.
3. **xorKey:** The key used to perform the XOR encryption and decryption.



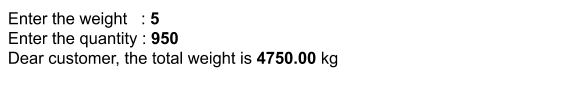
1. The code snippet above handles the XOR-ing of the user input and compares it with the characters stored inside strPw.

## *2.2 Four-digit number display*

One of the advanced features from the program is the 4-digit display feature. The main purpose of this feature is to support for realistic, large numbers, at least enough to handle LOH's daily sales needs. For instance, some customers may buy the meats in bulk, and therefore total weight of the meat purchased can reach up to 4 digits. As a result, the total price of the meat will also be up to 5 digits.

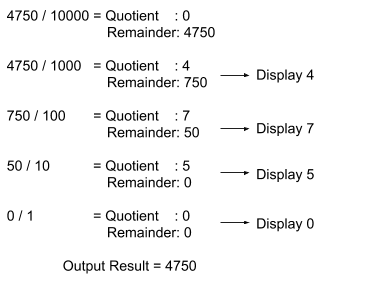
### 2.2.1 Operation Example

The user first enter the weight of meat (per packet) and quantity purchased. The total weight of meat (result) is then obtained through the multiplication of the weight of meat (per packet) and quantity.



*Figure : Input data by the user and the display of output result*

Suppose that the total weight of meat is 4750.00 kg. The result obtained will undergo a division process in order to be displayed.



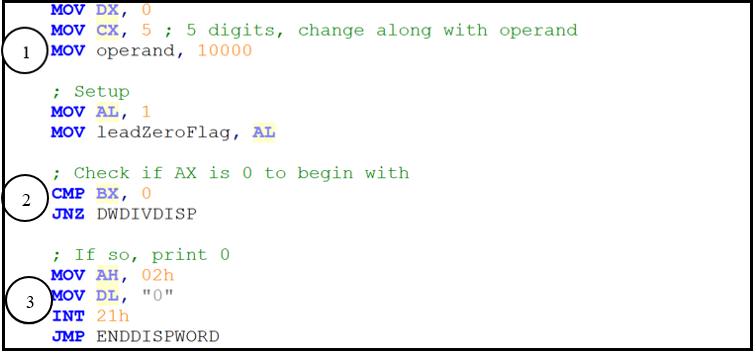
*Figure : Division process that is involved in displaying 4-digit number*

By applying the division process above, the 4-digit number can be displayed clearly and correctly to the user.

### 2.2.2 Algorithm

1. The number 10000 will first be assigned to a variable called “operand”.
2. The program will then divide the result (total weight of meat) by “operand”
3. After the division, if the quotient consists of any non-zero number, the quotient will be displayed. At the same time, the remainder will be stored for further division process.
4. Each time after the division of total weight by operand, the operand will automatically be divided by 10 (10000 → 1000 → 100 → 10 → 1)
5. The remainder will then be used to divide by the new operand and the quotient obtained will be displayed to the user.
6. The same division process is repeated until the remainder obtained is 0.

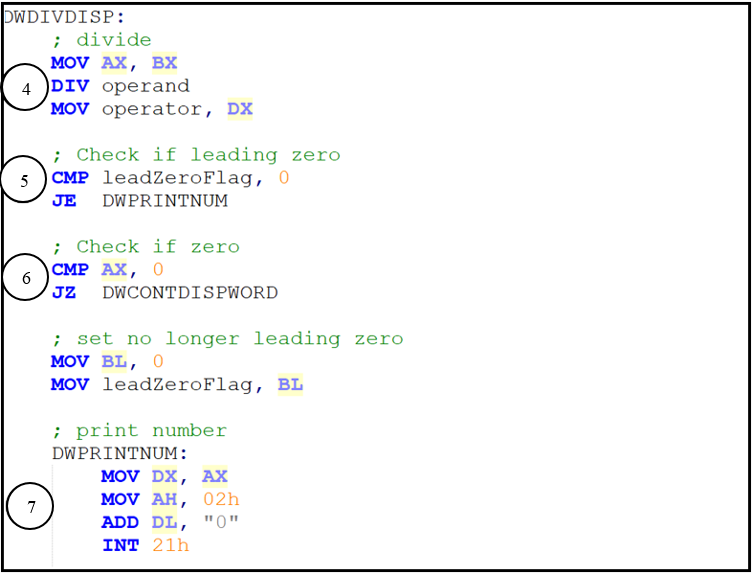
### 2.2.3 Code Snippet

https://docs.google.com/drawings/u/1/d/sbDsDD6hk-SRIDjGpCFwnkA/image?w=39&h=29&rev=2&ac=1&parent=1-Mve0YjxBPIW0icgrE1uySRPag6-AP1VYA6R_yKkIHE

1. **Operand**: This variable stores the initial value of 10000.

2. The program will check whether the total weight is 0kg or not.

3. If yes, the output result (total weight) will be displayed immediately without going through   
 any complicated division process.

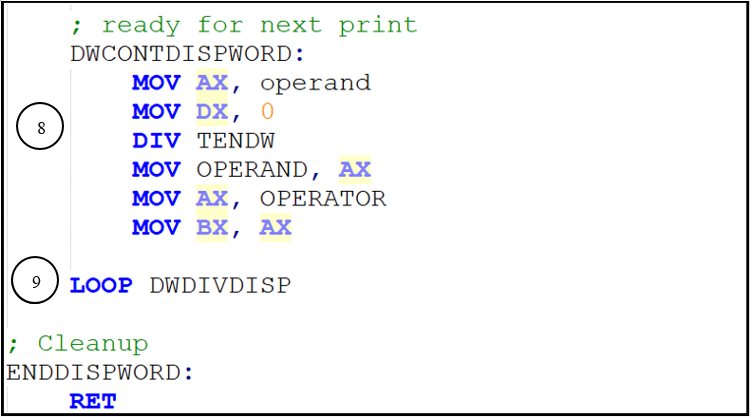


4. The total weight is divided by the operand.

5. Check leading zero. If so, nothing will be displayed.

6. Compare the quotient with 0. If they are equal, the program will carry out another process   
 (In the next screenshot).

7. If the quotient is not equal to 0, the number will be printed out and displayed.



8. The operand will divide itself by ten (From initial value of 10000 to 1000, from 1000 to   
 100, etc) continuously. [Continue from 5]

9.The division process is repeated in which the remainder will continue to divide itself by the   
   operand and display the number (quotient). It will then stop once the operand becomes ‘1’   
 and the remainder becomes ‘0’.

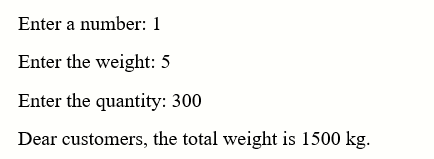
## *2.3 Five-digit number calculation*

Another advanced function in the program is the 5-digits number calculation. The main purpose is to support large input number from the user and is able to calculate numbers up to 5-digits. For example, calculation such as total weight \* price of meat and currency calculations such as total price/currency will involve calculations up to 5-digits but with a limitation of 2 byte which is 65535.

Therefore, calculations such as multiplication to obtain the total price of meat, subtraction to calculate the change for user and division to get a currency exchange can be done up to 5-digits.

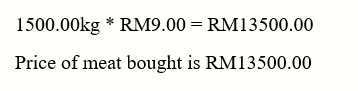
### 2.3.1 Operation Example

The user first enter the weight of meat (per packet) and quantity purchased. The total weight of meat (result) is then obtained through the multiplication of the weight of meat (per packet) and quantity.



*Figure : Input data by the user in the PURCHASE function*

Next, when the user enters the BILLING function, it will display an equation of weight of meat(kg) \* price of meat per kg(RM) = total price of meat purchased(RM). It will also display a statement “Price of meat bought is RM…...”.



*Figure : Display of output result in BILLING function*

The calculation of the figure are all done automatically after user input and it will be stored in the variable. For example,

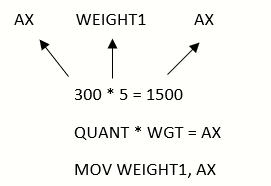


Figure : Multiplication calculation to obtain the total weight input by user

Next, the user will continue to input the choice for currency and the amount of meat bought after currency exchange will be displayed. After user enters the amount for payment, a message “Thank you! Your balance is USD…..”

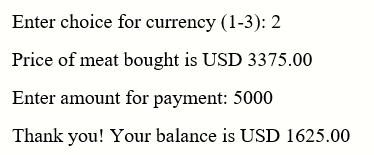
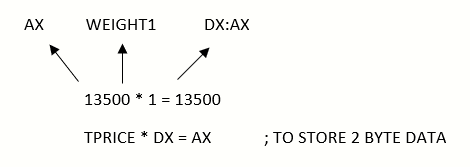


Figure : Input data from user for payment and the result

The calculation of the figure are all done automatically after user input and it will be stored in the variable. For example,



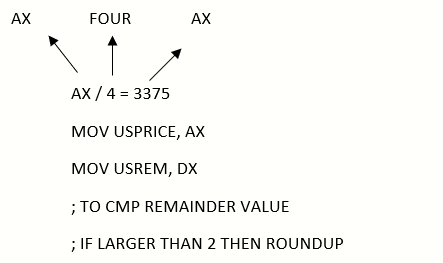


Figure : Division calculation to obtain the total price of meat bought after currency exchange

### 2.3.2 Algorithm

* Accept input string and convert it into digits

1. Accept input from the user
2. Check if it is 1 number, if it is one number it will store into BL and subtract 30H to get the real input value and lastly stored into the variable declared
3. If the input has more than 1 number, it will enter a loop.
4. The loop will store the 1st number into BL and subtract 30H then multiply with 10.
5. The loop will end when it reached the last number input.
6. After the loop calculation, the value will be stored in a variable declared.

* Multiplication calculation

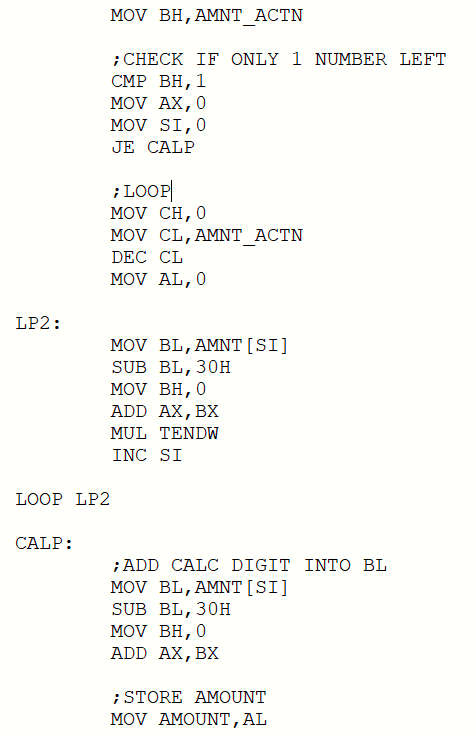
1. Move the multiplier into AX and multiply it with a multiplicand that must be a defined word to support 2 byte data.
2. Store the data into a defined word variable for display or continue with calculation

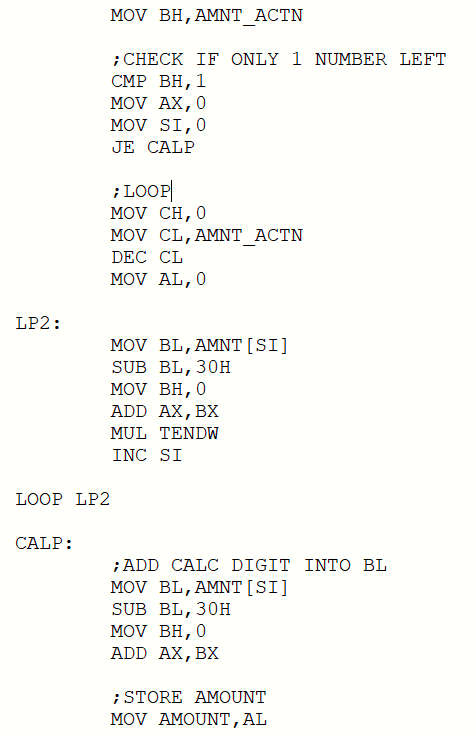
* Division calculation

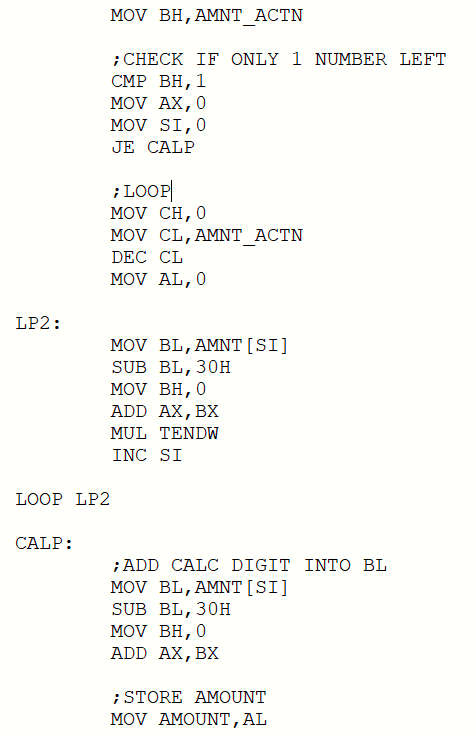
1. Move the dividend into AX and move 1 into DX
2. Multiply AX with DX for calculation of 4 byte data to be divided with 2 byte data to obtain 2 byte data result.
3. After that, divide AX with divisor and obtain the quotient value from AX
4. Lastly, compare the remainder value from DX and compare it value and if it is bigger than the value, it will round up the quotient and store it in a variable

### 2.3.3 Code Snippet

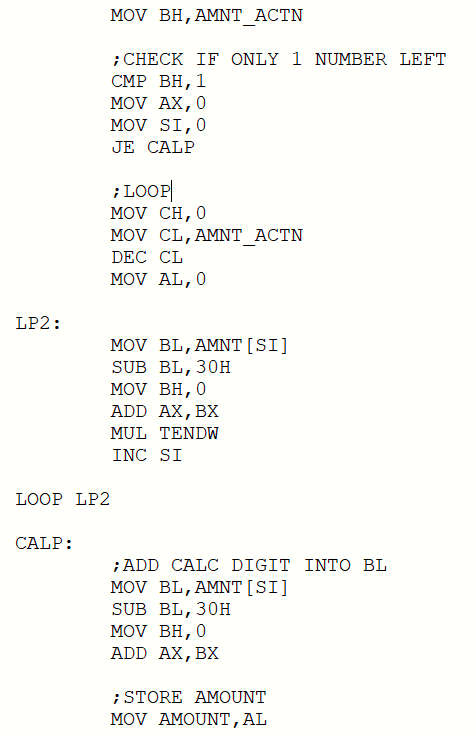
When user inputs digits into the program, the program will accept the input as string and it will convert the input digit string into digit by using a formula.







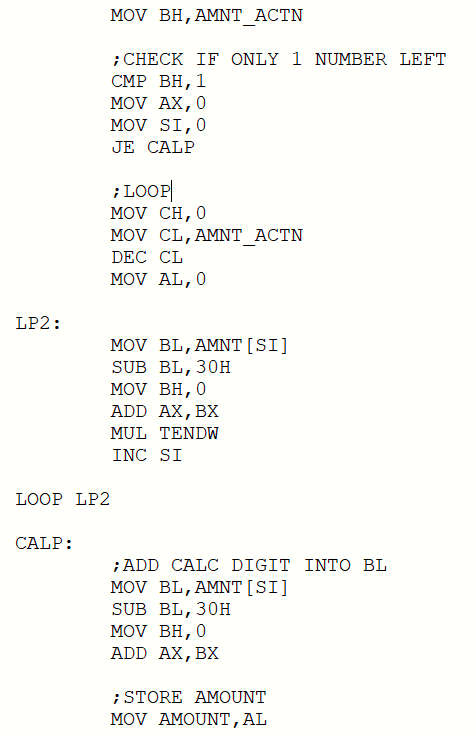
1. The formula will first check the number of input, if it is one digit it will convert it to digit by   
 subtracting 30H and store it in the variable declared.



2. But if the input digits are more than 1, it will enter a loop and it will looped based on the actual   
 number of digits input.

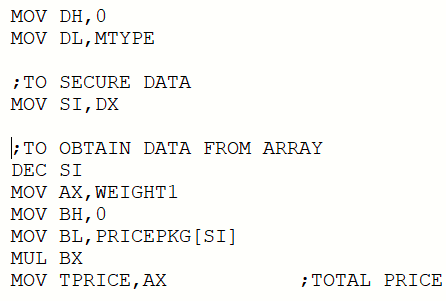
3. The loop will first subtract 30H from the first digit and it will be stored in BX and the value will be   
 moved to AX to be multiplied with 10.

4. The loop will continue with the calculation until the digits are all stored in AX



5. After converting the value to digits, the value will be stored in the variable declared for example   
 AMOUNT.

After user entered data in the purchase function, it will continue with the calculation to print the total price of meat.

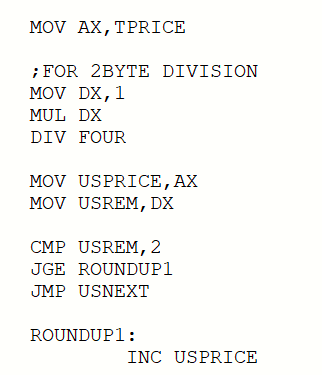


1. Data is stored in MTYPE and WEIGHT1 when the user input data in the purchase function

2. The value in MTYPE will be moved to SI and the value in SI will be decreased to obtain the value   
 of price of meat that is stored in the array.

3. The program will then multiply the WEIGHT1 and PRICEPKG[SI] to obtain the total price of meat   
 which can be up to 2 bytes.

Currency exchange calculation if the user wish to pay with another currency such as USD or SGD.



1. **TPRICE** which is the total price of meat bought is moved to **AX** to continue with the division.

2. **AX** is multiplied with **DX** to allocate enough space if **TPRICE** is a 2 byte number.

3. It is then divided by **FOUR** which is the currency rate for **USD**.

4. The quotient will be stored in **AX** and we move it to **USPRICE** to protect data while the remainder   
 will be stored in **DX** and we move it to **USREM** to protect the data as well.

5. Comparison of **USREM** and 2 is to round up the **USPRICE** if the remainder is larger or equal to 2.

# Part 3 I/O Design

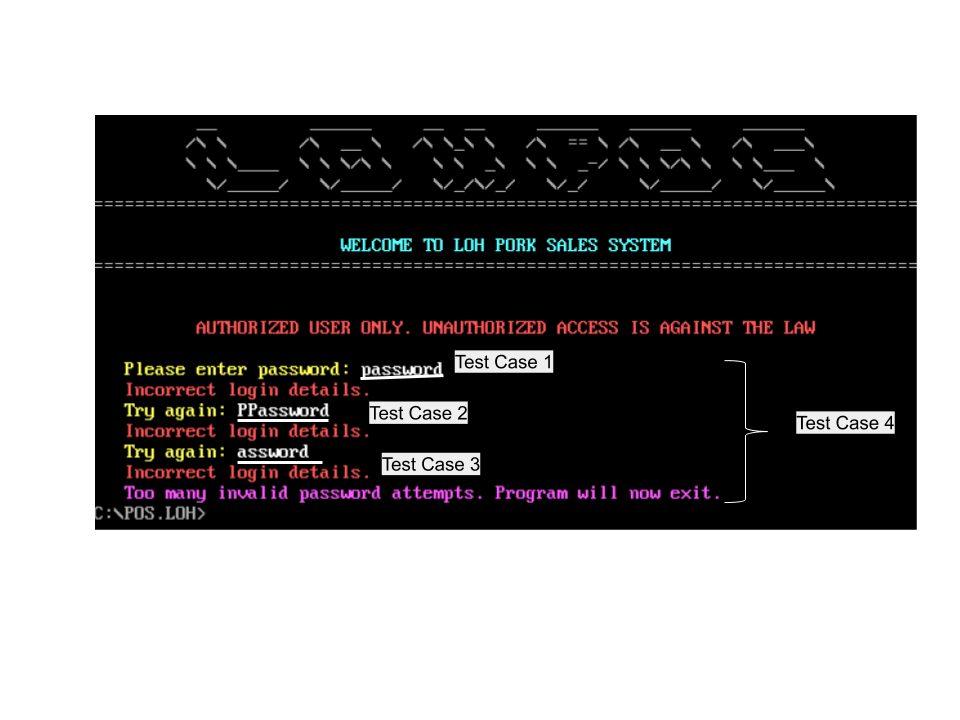
## *3.1 Login System*

**Purpose**: Authenticates & verifies user before allowing access to the system.

### 3.1.1 Sample Data

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Condition | Input | Expected & Obtained Result |
| 1 | Did not match password completely (symbols, numbers, capitalization) | password | Deny user entry and reprompt for password |
| 2 | Password longer than stored | PPassword | Deny user entry and reprompt for password |
| 3 | Password shorter than stored | Passwor | Deny user entry and reprompt for password |
| 5 | Too many attempts (by default,exceed 3 attempts) | password  PPassword  assword | Program terminates. |
| 4 | Password matches the stored password exactly | P@ssw0rd | Allow user entry and redirect user to main menu |

### 3.1.2 Screenshots & Program Execution



***Screenshot****: Execution of the login function, and test cases associated (Login success not shown, user redirected to main menu screen immediately)*

## *3.2 Main Menu System*

**Purpose:** Allow user to select the function they intend to use.

### 3.2.1 Sample Data

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Condition | List of inputs (separated by commas) | Expected & Obtained Result |
| 1 | Enters a valid choice | 1,2,3,0 | 1-3 Redirects the user to the respective module.  0: Displays summary, then quit the program |
| 2 | Enters an invalid choice | -,4,5,a,! | Reprompt for valid choice |

### 3.2.2 Screenshots & Program Execution

**Note:** On invalid choice, the screen automatically refreshes instead of filling the screen with error messages. Therefore, unless the user enters a valid choice, no visible feedback can be observed (except for occasional flashes of screen)



***Screenshot****: Operation when user enter ‘3’, a valid input corresponding to summary function*.

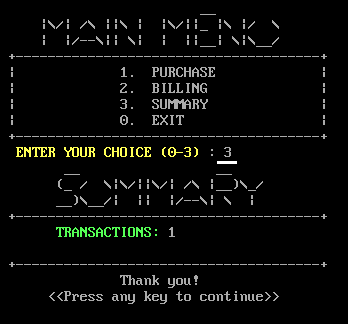
## *3.3 Summary System*

**Purpose:** Allow user to view the successful transactions made during the operation of the system.

### 3.3.1 Sample Data

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Condition | Input steps (, separated by ‘→’ ) | Expected & Obtained Result |
| 1 | User did not complete transaction | From main menu:  Enter summary:  3 → [Enter]  Enter purchase:  → 1 → 1 →  1 → 1 → N → [Enter]  Enter billing:  Enter summary:  → 3 → [Enter] | ‘Transactions’ figure  in summary do not change (because no valid transaction made yet) |
| 2 | User performs a valid transaction (Purchase + Billing), then enters summary | From main menu:  Enter summary:  3 → [Enter]  Enter purchase:  → 1 → 1 →  1 → 1 → N → [Enter]  Enter billing:  → 2 → 1 → 10 → [Enter]  Enter summary:  → 3 → [Enter] | ‘Transactions’ figure  in summary increased by 1 |

3.3.2 Screenshots & Program Execution



***Screenshot****: Operation when user enters ‘3’ in main menu, a valid input corresponding to summary function, after a successful transaction is made.*

## 

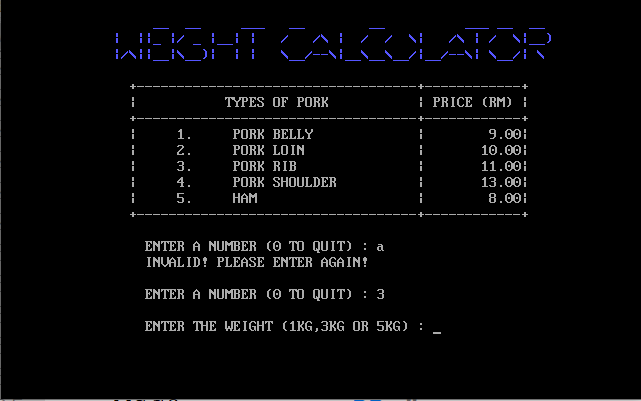
## *3.4 Purchase System*

**Purpose:** To calculate the total weight of meat purchased by the customers. The program will first read the weight of meat (per packet) and quantity entered by the customer and perform calculation. The total weight of meat is obtained through the multiplication of weight of meat (per packet) and the quantity entered.

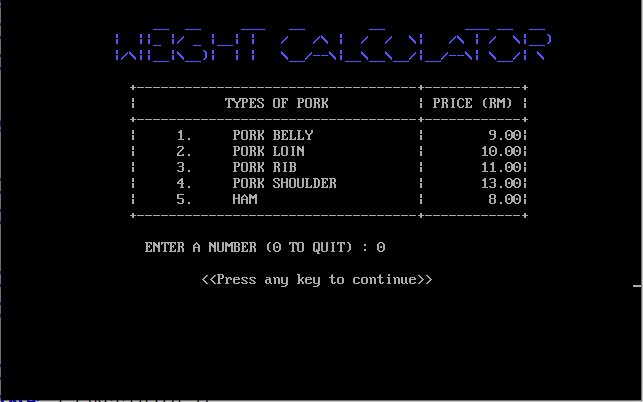
### 3.4.1 Sample Data

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Condition | Input data | Expected & Obtained Result |
| 1 | User provides an invalid input | Enter a number (0 to exit): **A, a, !, @, [Enter]** | An error message will be displayed and the user is required to re-enter a valid choice |
| 2 | User enters number ‘0’ | Enter a number (0 to exit):  **0** | User will quit the purchase function and will be redirected to the menu |
| 3 | User provides valid input | Enter a number (0 to exit):  **1, 2, 3, 4, 5, 6** | The user will then be prompted to enter the weight of meat (per packet) |
| 4 | User provides an invalid input | Enter the weight:  **2, 4, !, #, A, [Enter]** | An error message will be displayed and the user is required to re-enter a valid weight |
| 5 | User provides valid input | Enter the weight:  **1, 3, 5** | The user will proceed to enter the quantity purchased |
| 6 | User provides an invalid input | Enter the quantity (max 3 digits):  **A, 12A, !!@** | An error message will be displayed and the user is required to re-enter a valid quantity |
| 7 | User provides valid input | Enter the quantity (max 3 digits):  **1, 99, 988** | The input data is accepted and the total weight of meat will also be displayed |
| 8 | User provides invalid input | Enter a number (1-2):  **3, 5, @, A** | The user will be prompted to provide a valid data |
| 9 | User provides valid input | Enter a number (1-2): **1, 2** | The user can then edit the weight or quantity |

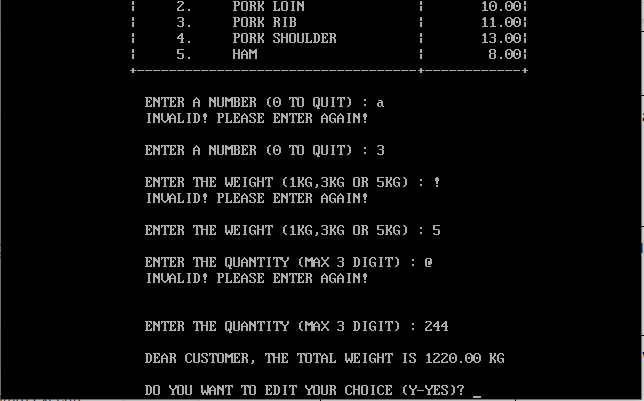
### 3.4.2 Screenshots & Program Execution



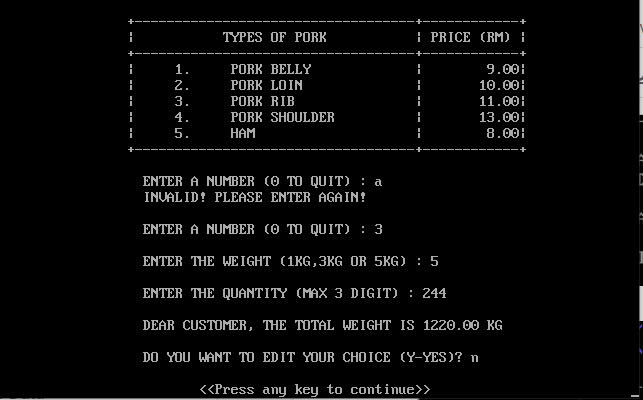
***Screenshot****: Test Case 1, 3*



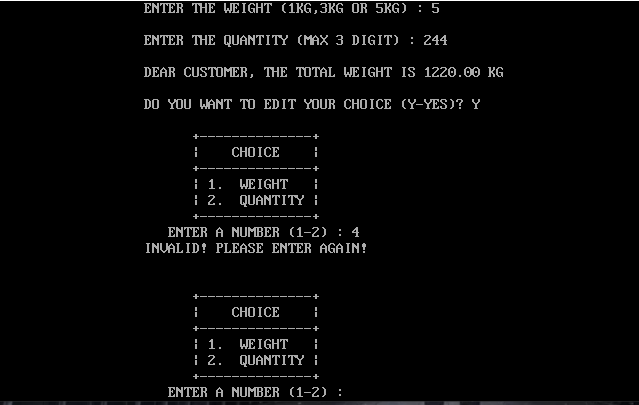
***Screenshot****: Test Case 2*

**

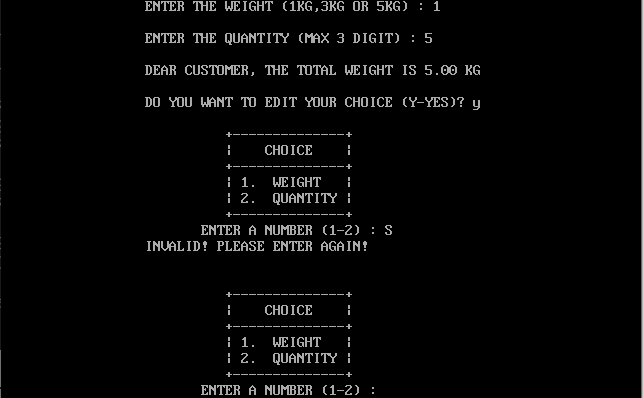
***Screenshot****: Test Case 1, 4, 5, 6*



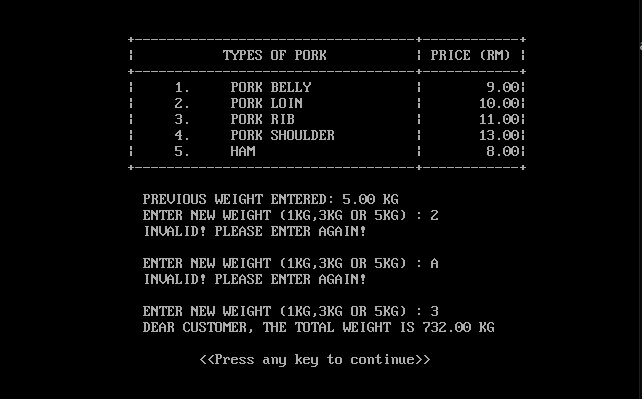
***Screenshot****: Test Case 7*



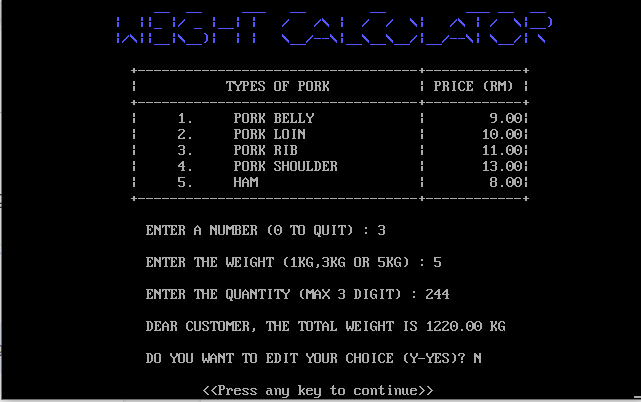
***Screenshot****: Test Case 8*



1



***Screenshot****: Test Case 9*



***Screenshot****: The user is able to view the total weight of meat if all the input data are valid*

## 

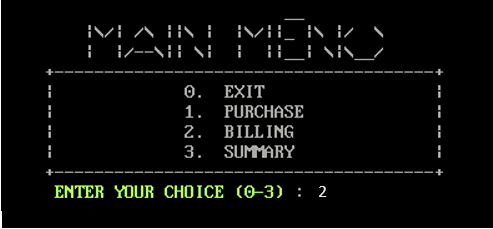
## *3.5 Billing System*

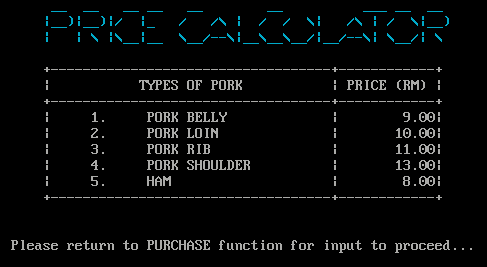
**Purpose:** To calculate the total price of meat purchased by the customers. If user enters the billing system without entering data in the purchase system, the program will prompt a message to inform the user to perform weight calculation before entering the billing system. If all the data are entered in the purchase system, an equation of price of meat \* weight of meat = total price of meat bought will be displayed. Lastly, the program will then prompt user for the currency they wished to continue with and the user will have to enter the amount for payment.

### 3.5.1 Sample Data

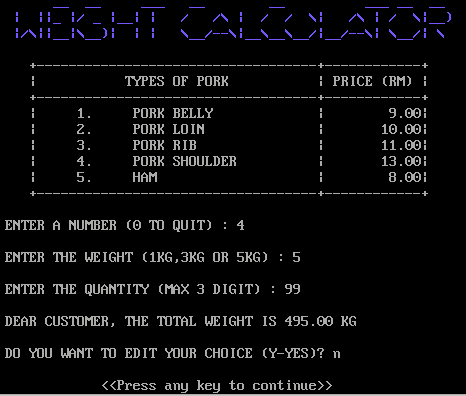
|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Condition | Input data | Expected & Obtained Result |
| 1 | User did not enter data in the purchase system | From main menu:  Enter billing:  2 → [Enter] | A prompt message “Please return to PURCHASE function for input to proceed” will be displayed |
| 2 | User provides an invalid input | Enter choice for currency(1-3): **A, - , !, @, [Enter]** | An error message will be displayed and the user is required to re-enter a valid choice |
| 3 | User provides valid input | Enter choice for currency(1-3):  **1, 2, 3** | The price of meat will be displayed and the user will then be prompted to enter the amount for payment |
| 4 | User provides an invalid input | Enter amount for payment:  **-, !, #, A, [Enter], Amount less than price** | An error message will be displayed and the user is required to re-enter a valid amount |
| 5 | User provides valid input | Enter amount for payment:  **Amount more than price** | The system will automatically calculate the change and a message “Thank you! Your balance is ….” will be displayed |

3.5.2 Screenshots & Program Execution



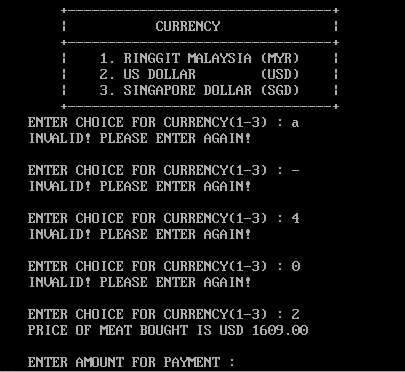


***Screenshot****: Test Case 1*

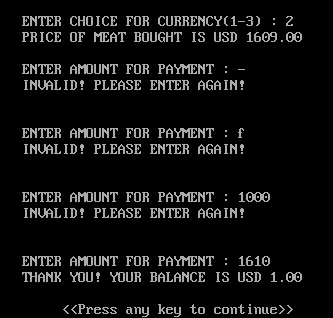




***Screenshot****: After user input data in the purchase function, they are able to view the weight, price of type of meat entered and the total price of meat bought in the billing function*



***Screenshot****: Test Case 2,3*

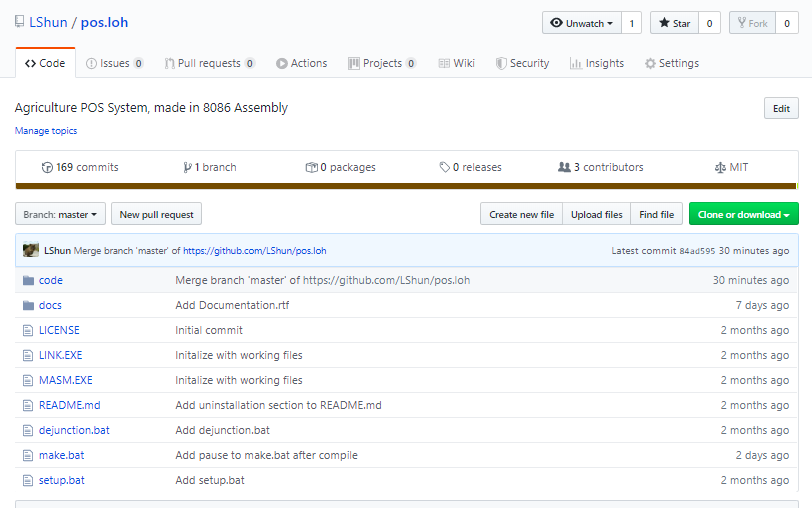
**

***Screenshot****: Test Case 4,5*

# Part 4 Leadership and Teamwork Software used

## *4.1 Git VCS*

During the project period, our group shared & regularly communicated our progress using Git Version Control System (VCS) through an online platform known as GitHub. By using GitHub, we successfully eliminated the need for "combining work" and could instead focus on developing our own part while at the same time ensure our work can function correctly alongside other member's functions. Besides that, we can also monitor and help out each other through Git, and roll back unwanted changes if corruption, or mistakes occur.



## *4.2 WhatsApp*

During the project period, our group utilized WhatsApp to communicate regarding our project progress and request for help. We used WhatsApp to notify each other on deadlines, and communicate ideas on our project that does not involve code.

# 

# Part 5 References

**GeeksforGeeks**. 2019. XOR Cipher - GeeksforGeeks. [ONLINE] Available at: https://www.geeksforgeeks.org/xor-cipher/. [Accessed 27 December 2019].

**Wikipedia**. 2020. INT 10H - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/INT\_10H. [Accessed 02 January 2020].

**XOR Encryption**. 2020. XOR Encryption. [ONLINE] Available at: https://www.tech-faq.com/xor-encryption.html. [Accessed 02 January 2020].

assembly?, H. and Rodríguez, J. (2020). *How to convert String to Number in 8086 assembly?*. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/36979870/how-to-convert-string-to-number-in-8086-assembly [Accessed 5 Jan. 2020].

# 

# Part 6 Appendices

Appendix 6.1: How XOR cipher works by The Tech-FAQ

