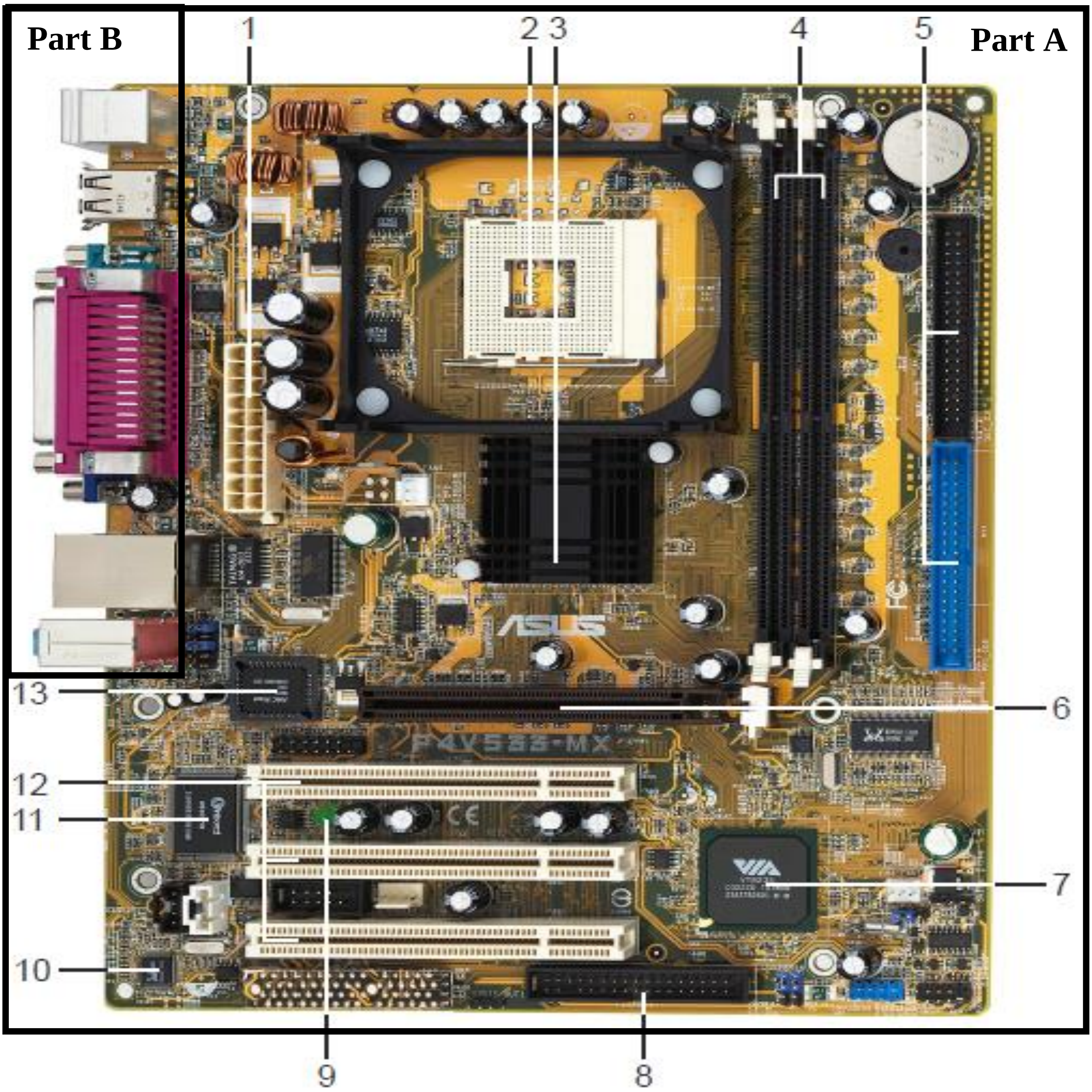
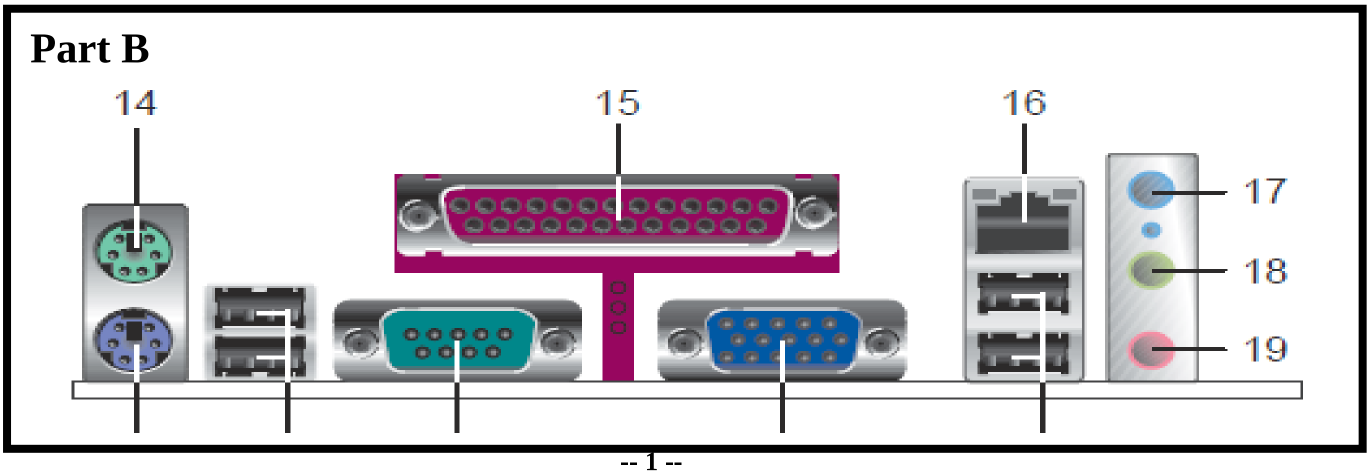
Practical 1: Hardware of a Computer System

Match the labels in **Part A** and **Part B** with the **Table 1** below.





Complete the table below:

No	Name	Description / Function	Category
	USB 2.0 ports 1 & 2	For connecting Universal Serial Bus (USB) 2.0 devices	Port
	AGP slots	Accelerated Graphic Port (AGP)	Slot
	Super I/O Controller	Support floppy disk drive, parallel port, Game / MIDI port	Controller
	CPU socket	A 478-pin surface mount, Zero Insertion Force (ZIF) socket for the Intel® 4 Procesor, with 533/400 MHz system bus that allows 4.2 GB/s and 3.2GB/s data transfer rates, respectively	Socket
	North bridge controller	To provide the processor interface, system interface, accelerated graphic port (AGP) interface and write protocol	Bus Controller
	PCI slot	Peripheral Component Interconnect (PCI) is a local computer bus for attaching hardware	Slot
	IDE connectors	Integrated Drive Electronics (IDE) is a standard interface for connecting a motherboard to storage devices	Connector
	Line in jack	(light blue) Connect a tape players / other audio sources	Jack
	Floppy disk connector	To accommodate the ribbon cable for the floppy disk drive	Connector
	DDR DIMM socket	Double data rate (DDR) dual in-line memory module (DIMM) is used to support system memory	Socket
	Audio CODEC	A device or computer program capable of coding or decoding a digital data stream of **audio**	Audio codec
	Serial port	For connecting pointing devices / other serial devices	Port
	South bridge controller	To provide the I/O interfaces	Bus Controller
	PS/2 mouse port	For a mouse connection (6 mini-DIN connector)	Port
	Power connector	A 20-pin connector connects to a power supply	Connector
	RJ-45 port	Allow LAN connection through a network hub	Port
	PS/2keyboard port	To connect to PS/2 keyboard	Port
	Line out jack	(line) Connect a headphone / a speaker	Jack
	Parallel port	For printer, scanner or other devices	Port
	Flash ROM	Contains programmable BIOS program	Memory
	VGA port	Video graphic adapter (AGV) for monitor	Port
	Standby power LED	A reminder to turn off the system power before plugging / unplugging devices	LED
	USB 2.0 ports 3 & 4	For connecting Universal Serial Bus (USB)2.0 devices	Port
	Microphone jack	(pink) Connects a microphone	Jack

Table 1: Hardware Components in System Unit

Practical 2: Using Debug Commands

Commands learnt:
A, C, D, E, F, G, H, P, Q, R, T, U commands

1. Use **DEBUG** program with appropriate command to perform the following related operations as shown in the **Table 1**.

	Command	Output
a)	`E CS:100 "This is to store ascii code to memory"`	
b)	`F CS:150 "This is to store ascii code to memory"`	

c)	`-a 200`
	`XXXX:0200 JMP 238`
	`XXXX:0202 DB 0D,0A,"IT'S OK TO RUN THIS "`
	`XXXX:0218 DB "PROGRAM UNDER DOS OR WINDOWS!"`
	`XXXX:0235 DB 0D,0A,24`
	`XXXX:0238 MOV DX,102`
	`XXXX:023B MOV AH,9`
	`XXXX:023D INT 21`
	`XXXX:023F MOV AX,4C01`
	`XXXX:0242 INT 21`
	`XXXX:0244`	

d)	`F CS:150 "This is to store ascii code to memory"`
	`S CS:100 13F "ascii"`	

Table 1

2. Use DEBUG program to perform the following **unrelated** operations serially and complete the **Table 2** below.

	Commands		**What do you see? Explain the content you get.** *Note: Copy and paste your commands & result onto this section.*
	Format	**Sample**	
1	**A**	**A**	
2	**A address**	**A 200**	
3	**C range address**	**C 100 105 200**	
4	**D**	**D**	
5	**D address**	**D 100**	
6	**D range**	**D 150 15A**	
7	**E address**	**E 100**	
8	**E address list**	**E CS:100 "Hello"**	
9	**F range list**	**F 100 13F, ' '**	
10	**F range, value**	**F cs:100 12F,FF**	
11	**F address size 'ASCII'**	**F 100 L 20 'Q'**	

12	**H operand1 operand2**	**H 2222 3333**	
13	**M range**	**M 100 108 170**	
14	**P**	**P**	
15	**P no_of_instruction**	**P 2**	
16	**P=address**	**P=100**	
17	**P=address,no_of_instruction**	**P=100,2**	
18	**R**	**R**	
19	**R register**	**R IP**	
20	**R category**	**R F**	
21	**S range list**	**S 100 110 "HELLO"**	
22	**T**	**T**	
23	**T no_of_instruction**	**T 2**	
24	**T=address no_of_instruction**	**T=100 2**	
25	**U**	**U**	
26	**U start_address**	**U 100**	
27	**U address_range**	**U 100 105**	

Table 2: Basic Debug Commands

3. Use **DEBUG** program with appropriate command to perform the following operations as shown in the **Table 3**.

	Operation	Commands
a)	Reset the IP address to **0707H.**	
b)	Insert the machine code below into memory code segment starts at offset address **0110H**. **B0 28 B3 3B 86 C3 D0 E0 FE C3 00 D8**	
c)	Display the content in memory segment to ensure that you have entered the object code correctly.	
d)	The object code **3BH** was supposed to be **3AH**. Correct it.	
e)	What are the symbolic instructions represented by the object code above?	
f)	Retrieve the initial content of all the register. Then, execute each instruction one by one.	

Table 3: Debug Operations

4. Using **DEBUG** program, identify and explain the value in **AB, BX, CX** and **DX** when the following instructions are executed as shown in **Table 4** below. (Note: Each set of instruction is run separately.)

Program	Working
a.) **A 100** **MOV AL, 30H ;AL = multiplicand** **MOV BL, 02H ;BL = multiplier (8 bits)** **MUL BL** **JMP 100H**	
b.) A 100 **MOV AX, 0083H ; AX = dividend** **MOV BL, 2H ; BL = divisor (8 bits)** **DIV BL** **JMP 100H**	
c.) **A 100** **MOV AX, 3000H ;AX = multiplicand** **MOV BX, 200H ;BX = multiplier (16 bits)** **MUL Bx** **JMP 100H**	
d.) A 100 **MOV AX, 8003H ; AX = dividend** **MOV BX, 100H ; BX = divisor (16 bits)** **DIV BX** **JMP 100**	

Table 4: Arithmetic Computation

5. Assume that **AL** contains 3A$_{16}$ and that an item named **BL** contains **C5**$_{16}$. Using **DEBUG** program, determine the effect on **AL** for the following **unrelated** operations by using debug program.

	Commands	Debug Program	Working
a)	**AL • BL**		
b)	**AL + BL**		
c)	**AL’**		
d)	**AL ⊕ BL**		

Table 5: Bitwise Logical Operation

Practical 3: Assembly Language Program

Instructions learnt:

MOV, MOVZX, ADD, SUB, MUL, DIV, SHL, SHR, INC, DEC, JMP, CMP, Jnnn, XCHG, AND, OR, XOR, NOT, CBW, LOOP, etc.

1. Display symbol

Write an assembly language program to display a smiley face (☺). The modify your program to show more symbols in the new line

Sample output:

```
☺
1 + 2 = 3
```

2. Simple I/O

Write an assembly language program to prompt user to enter a character from A - J. Then, write a formula to change uppercase to lowercase.

Sample output:

```
Enter an uppercase (A - J): H
The lowercase of the character entered is: h
```

3. Arithmetic Calculation: Multiplication & Division

Write a program that prompts the user to enter the quantity and unit price

(a) Single digit input. Then, display the total amount (single digit) with an appropriate message.

Sample output:

```
Quantity (unit): 3
Unit price (RM): 2
Total amount is RM6
```

(b) Single digit input. Then, display the total amount (2 digits) with an appropriate message

Sample output:

```
Quantity (unit): 8
Unit price (RM): 5
Total amount is RM40
```

(c) Two digits input. Then, display the total amount (3 digits) with an appropriate message

Sample output:

```
Quantity (unit): 15
Unit price (RM): 7
Total amount is RM105
```

4. Indirect addressing: `LOOP` instruction

Given the following data items, write an assembly language program to perform the followings.

```
VAR1 DB "COMPUTER"
VAR2 DB 8 DUP ('*')
```

(a) Convert uppercase to lowercase.

Sample output:

```
Before convert case
-------------------
VAR1 = COMPUTER
VAR2 = ********

AFTER convert case
------------------
VAR1 = COMPUTER
VAR2 = computer
```

(b) Reverse the content of **`VAR1`** and store it into **`VAR2`**. Display the output.

Sample output:

```
Before reverse
--------------
VAR1 = COMPUTER
VAR2 = ********

AFTER reverse
-------------
VAR1 = COMPUTER
VAR2 = RETUPMOC
```

5. Find the sum and average: Using `LOOP` and `CMP` instruction

Given the data definition as followed, write an assembly language program to calculate the sum and the average.

```
NUM DB 6, 12, 7, 5, 9, 11, 13
```

Sample output 1:

```
SUM: 63
Ave: 9.0
```

Sample output 2:

```
SUM: 87
Ave: 9.6
```

(a) Use **`LOOP`** instruction to complete the above program. Refer to **Sample Output 1**.

(b) Rewrite your program using **`CMP`** instruction. Refer to **Sample Output 1**.

(c) Assume that `NUM` has been added with additional two new values, i.e. **16** and **8**. What is the new `SUM` and `AVE` respectively? Refer to **Sample Output 2**.

6. Validation: Security login & digit validation

(a) Security login

Given the following data definition, write an assembly language program to perform validation for security login.

```
UNAME DB “ICS”
UPSW  DB “1024”
```

Sample output:

```
Username: ICA
Password:
Invalid login!
Username: ICS
Password:
Access Granted!
```

(b) Digit validation

Refer to Question 3(b), perform the digit validation for user input.

Sample output:

```
Quantity (unit): 8
Unit price (RM): m
Invalid input! Enter digit only!
Unit price (RM): 5
Total amount is RM40
```

Self-review:

7. Create an Assembly Language program that find the first even number in an array of unsigned integers: `79, 93, 30, 4, 6, 1.`

8. Create an Assembly Language program to print the number `190, 5, 37, 66,` and `4` on the screen:

(a) using `LOOP` instruction.

(b) Using `CMP` instruction.

Practical 4: Unix OS

Commands learnt:
sudo, su, date, cal, whoami, users, who, id, w, uname, man, info, help

1. Write the output for date command using each of the following format:

Commands	Result
(a) `date " +This is the date"`	
(b) `date "+This is the date: %B"`	
(c) `date "+Date: %d %B %Y"`	
(d) `date "+Today is %A (%d/%m/%Y)"`	
(e) `date "+Date Today: %d-%m-%Y & Time Now: %l:%M %p"4a. 4b.`	

2. Write command(s) based on the following scenario:

Functions	Commands
(a) To display the manual for the `sed` command	
(b) To check the name of the Linux system	
(c) To clear terminal screen and exit the terminal	
(d) To view for the current month calendar	
(e) To show the summary of gzip command	

3. Write command(s) based on the following scenario:

Function	Note
(a) To list the date and time of the server.	
(b) To set time and date to the value specified. Eg. "11/20/2019 12:48:00"	
(c) To show calendar of the specified month and year.	
(d) To provide in depth information about the requested command.	
(e) To search for commands related to a particular keyword.	
(f) To display a short help file.	
(g) To clear the screen.	
(h) To displays the name of the current working directory.	
(i) To become super user or another user.	
(j) To allow a user to execute a command as the superuser.	
(k) To leave the current shell session.	

Practical 5: Basic Unix File Implementation

Commands learnt:
ls, mkdir, cd, pwd, cat, head, tail, tac, more, less, vi, rm, cp, mv, grep.

1. Perform the following operations:

Operations	Commands
1) Show the current path.	
2) Create a directory "p2".	
3) Give two commands which can list all the contents at the current directory.	
4) Create a directory "super" inside the directory "p2".	
5) Inside this "super" directory, create four files "apple.doc", "april.doc", "box.txt" and "boxes.txt" using touch command.	
6) (a) Show the file(s) starting with "ap". (b) Show the file(s) that the length of the filename contains 5 characters. (c) Show the file(s) ending with "txt" extension. (d) Show the file(s) that the third character of the filename is "p".	
7) Go to root directory.	
8) Search a directory named "super" from the root directory.	
9) Search a file named "ultra.txt" from the root directory.	
10) Rename the file "ultra.txt" to "mega.txt"	
11) Copy "super" directory to home directory and rename it to "super2" in one command line.	
12) Confirm whether "super2" directory is created and copied.	
13) Remove "super" and all its subdirectories and files.	
14) Go back to directory "p2". Create a text file "sampleA.txt" using vi command	
15) Type the following contents into "sampleA.txt" and save the file. "The shell is a program that takes keyboard commands and passes them to the operating system to carry out. Almost all Linux distributions supply a shell program from the GNU Project called bash."	
16) Create another text file "sampleB.txt" using cat command.	
17) Type the following contents into "sampleB.txt" and save the file. "When using a graphical user interface, we need another program called a terminal emulator to interact with the shell."	
18) Open "sampleB.txt" again using vi command and add the following sentences.	

`“It’s likely called simply “terminal”.”`	
19) Show the word count of “`sampleB.txt`”	
20) Combine the two files “`sampleA.txt`” and “`sampleB.txt`” to a file named “`sampleC.txt`”.	
21) Show the content of “`sampleC.txt`” on the terminal.	
22) Archive “`sampleA.txt`” and “`sampleB.txt`” to “`sampleC.tar`”	
23) Overwrite “`sampleB.txt`” by “`sampleA.txt`” and show the content of both files on the terminal.	
24) Delete the file “`sampleB.txt`”.	
25) Go back to home directory and create a new directory named “`backup`”.	
26) Backup “`sampleA.txt`” and “`sampleC.txt`” to “`backup`” directory.	

2. Write commands for the following instructions by using the vi editor:

a) Create a document called `first.file`, and enter 4 lines of words “`today`” and save it.

b) Copy the first 4 lines using only one command

c) Save the file, and exit `vi`

d) Create a second document called `second.file`, and enter 4 lines of words “`yesterday`” and save it.

e) Create a third document called `third.file`, by merging the text from the first two files.

f) Save the `third.file` and exit from the editor

Practical 6: Korn Shell

Commands learnt:

Cat, sudo, eco, grep, chsh, sort, cut, sed, tr, chmod,

1. Perform the following operations:

Command / Shell Script		Result

a)	`echo -e “What is your favorite color?--> \c”`
	`read REPLY`
	`   if [ “$REPLY” = “red” –o “$REPLY” = “blue” ]`
	`then`
	`   echo “The answer is red or blue.”`
	`else`
	`   echo “The answer is not red nor blue.”`
	`fi`	

b)	`file /usr/bin/* | grep "Again shell script" | sort -r`	
c)	`tr a A < /etc/hosts | sort -r | pr -d > /etc/hosts`	

d)	``for file in `ls *.txt` ``
	`do`
	`  echo -n “Display $file? “`
	`  read answer`
	`  if [ $answer == ‘y’ ]`
	`  then`
	`     less $file`
	`  fi`
	`done`	

2. Write a script that takes the name of a directory as an argument and searches the file hierarchy rooted at that directory for zero-length files. Write the names of all zero-length files to standard output. If there is no option on the command line, have the script delete the file after displaying its name, asking the user for confirmation, and receiving positive confirmation. A **–f** (force) option on the command line indicates that the script should display the filename but not ask for confirmation before deleting the file.