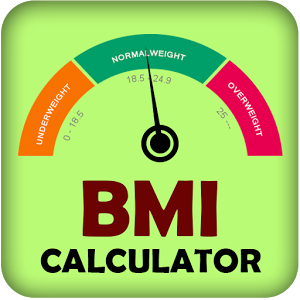
**PROJECT REPORT**

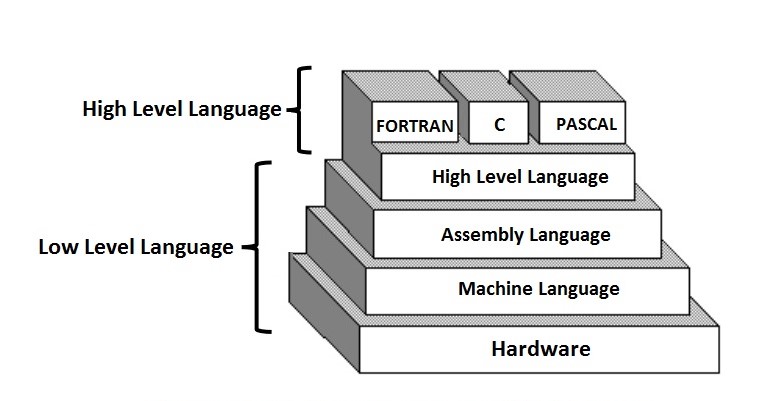
**Project Name: BMI CALCULATOR**

**BMI CALCULATOR**

****

**Introduction:**

**An assembly language is a low-level programming language for microprocessors and other programmable devices. Intermediate-level programming language which is higher (is easier to use but runs slower) than machine language and lower (is more difficult to use but runs faster) than a high-level language such as Basic, Fortan, or Java. Programs written in assembly language are converted into machine language by specialized programs called assemblers or compilers for their execution by the machine (computer).**

****

**A program written in assembly language consists of a series of (mnemonic) processor instructions and meta-statements (known variously as directives, pseudo-instructions and pseudo-ops), comments and data. Assembly language instructions usually consist of an**[**opcode**](https://en.wikipedia.org/wiki/Opcode)**mnemonic followed by a list of data, arguments or parameters. These are translated by**[**assembler**](https://en.wikipedia.org/wiki/Assembly_language_assembler)**into**[**machinelanguage**](https://en.wikipedia.org/wiki/Machine_language)**instructions that can be loaded into memory and executed.**

**For example, the instruction below tells an**[**x86**](https://en.wikipedia.org/wiki/X86)**/**[**IA-32**](https://en.wikipedia.org/wiki/IA-32)**processor to move an**[**immediate 8-bit value**](https://en.wikipedia.org/wiki/Constant_(programming))**into a**[**register**](https://en.wikipedia.org/wiki/Processor_register)**. The binary code for this instruction is 10110 followed by a 3-bit identifier for which register to use. The identifier for the AL register is 000, so the following**[**machine code**](https://en.wikipedia.org/wiki/Machine_code)**loads the AL register with the data 01100001.**

**10110000 01100001**

**This binary computer code can be made more human-readable by expressing it in**[**hexadecimal**](https://en.wikipedia.org/wiki/Hexadecimal)**as follows.**

**B0 61**

**Here, B0 means 'Move a copy of the following value into AL', and 61 is a hexadecimal representation of the value 01100001, which is 97 in**[**decimal**](https://en.wikipedia.org/wiki/Decimal)**. Assembly language for the 8086 family provides the**[**mnemonic**](https://en.wikipedia.org/wiki/Mnemonic)[**MOV**](https://en.wikipedia.org/wiki/MOV_(x86_instruction))**(an abbreviation of move) for instructions such as this, so the machine code above can be written as follows in assembly language, complete with an explanatory comment if required, after the semicolon. This is much easier to read and to remember.**

**MOVAL, 61h; Load AL with 97 decimal (61 hex)**

**In some assembly languages the same mnemonic such as MOV may be used for a family of related instructions for loading, copying and moving data, whether these are immediate values, values in registers, or memory locations pointed to by values in registers. Other assemblers may use separate opcode mnemonics such as L for "move memory to register", ST for "move register to memory", LR for "move register to register", MVI for "move immediate operand to memory", etc.**

**The x86 opcode 10110000 (B0) copies an 8-bit value into the AL register, while 10110001 (B1) moves it into CL and 10110010 (B2) does so into DL. Assembly language examples for these follow.**[**[8]**](https://en.wikipedia.org/wiki/Assembly_language#cite_note-intel-1999-MOV-8)

**MOVAL, 1h; Load AL with immediate value 1**

**MOVCL, 2h; Load CL with immediate value 2**

**MOVDL, 3h; Load DL with immediate value 3**

**The syntax of MOV can also be more complex as the following examples show.**[**[9]**](https://en.wikipedia.org/wiki/Assembly_language#cite_note-9)

**MOVEAX, [EBX] ; Move the 4 bytes in memory at the address contained in EBX into EAX**

**MOV [ESI+EAX], CL; Move the contents of CL into the byte at address ESI+EAX**

**In each case, the MOV mnemonic is translated directly into an opcode in the ranges 88-8E, A0-A3, B0-B8, C6 or C7 by an assembler, and the programmer does not have to know or remember which.**

**Transforming assembly language into machine code is the job of an assembler, and the reverse can at least partially be achieved by a**[**disassemble**](https://en.wikipedia.org/wiki/Disassembler)**. Unlike**[**high-level languages**](https://en.wikipedia.org/wiki/High-level_language)**, there is usually a**[**one-to-one correspondence**](https://en.wikipedia.org/wiki/One-to-one_correspondence)**between simple assembly statements and machine language instructions. However, in some cases, an assembler may provide pseudo instructions (essentially macros) which expand into several machine language instructions to provide commonly needed functionality. For example, for a machine that lacks a "branch if greater or equal" instruction, an assembler may provide a pseudo instruction that expands to the machine's "set if less than" and "branch if zero (on the result of the set instruction)". Most full-featured assemblers also provide a rich**[**macro**](https://en.wikipedia.org/wiki/Macro_(computer_science))**language (discussed below) which is used by vendors and programmers to generate more complex code and data sequences.**

**Each**[**computer architecture**](https://en.wikipedia.org/wiki/Computer_architecture)**has its own machine language. Computers differ in the number and type of operations they support, in the different sizes and numbers of registers, and in the representations of data in storage. While most general-purpose computers are able to carry out essentially the same functionality, the ways they do so differ; the corresponding assembly languages reflect these differences.**

**Multiple sets of**[**mnemonics**](https://en.wikipedia.org/wiki/Mnemonic)**or assembly-language syntax may exist for a single instruction set, typically instantiated in different assembler programs. In these cases, the most popular one is usually that supplied by the manufacturer and used in its documentation.**

**There is good news and bad news about studying assembly language programming. The bad news is that it takes too much effort to write applications in assembly language. No employer is going to pay you to write a large complex assembly language program.**

**Then why should you learn assembly language? I mean other than be able to graduate? That is the good news.**

* **In order to write high-level languages, such a C/C++ and Pascal, it is necessary to have some knowledge of the assembly language they translate into.**

**Programs written in high-level languages will usually not run as fast as assembly language programs. In a extremely small percentage of applications, speed is so critical that only assembly language routines can meet the speed requirements.**

* **Sometimes to debug a higher-level language, you have to review the resulting assembly language.**
* **Compiler writers must know how to write assembly language in order to have the compiler do code generation.**
* **Assembly language programmer can earn more than programmers who can not write assembly language (in those applications where assembly language is required).**
* **Writing assembly language is fun.**

**What you will encounter in the real-world is that there are a set of functions that are time-sensitive and will be written in assembly language. It is necessary that you learn how to mix assembly language and other languages.**

**Assembly language is used for transforming higher-level programming languages like C into machine code. Processors can only run machine code -- a sequence of short, discrete, instructions encoded in binary format. Every time any program runs, machine code is being executed by a processor.**

**Background:**

**In this project we use some new syntax and the details is given bellows-**

**INT 21h**

* **Here INT 21h is used for getting input.**

**AND AX, 000FH**

* **For converting the character into digit.**

**PUSH AX**

* **For keeping the value of AX into the stack.**

**POP AX**

* **For geting the value of AX from the stack.**

**MUL BX**

* **For multiplying the value of AX with BX.**

**CMP AL, 0DH**

* **For comparing the value of AL with Enter.**

**DIV BX**

* **For dividing the value of AX by BX.**

**We also use a WORD variable SUM.**

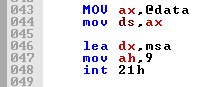
**Project Description:**

* **BMI is a way of checking that a person is a healthy weight for their height.**
* **For your height and weight there will be a healthy range which you should fit into, this range is to account for the fact people have different builds and sizes.**

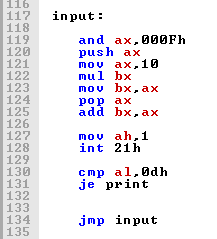
**In this project we input our height(in cm) and weight (in kg) and its show me the result that my weight is OVER or PERFECT or UNDER.**

**In this project basically we had done the work of PUSH, POP,MUL,DIV and other commanding thing like JMP, LOOP, and CMP etc.**

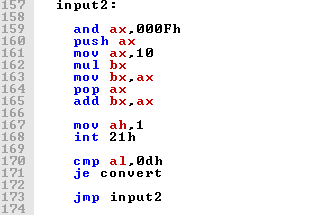
**Now we will discuss about the some important part of our project code.**

****

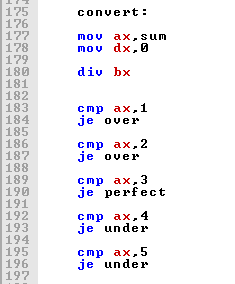
**The above picture is the part of our code, use for print any kind of massage. So we will use this command and print all massage in our project by using this part of code.**

****

**The above picture is the part of our code, used for any kind of decimal input that is HEGHT of the user in cm.**

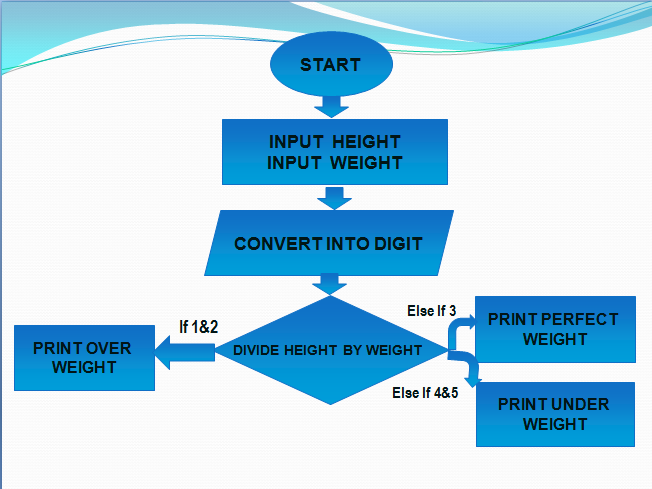
****

**The above picture is the part of our code, use for 2nd input that is the WEIGHT of the user in kg.**

****

**The above picture is the part of our code, use for conversion, that is the BMI of the user. In the last part of the program we show some instructions for the user. If He or She has Over weight then have some instructions and if He or She has Under weight then have some instructions.**

**Flow chart:**

****

**In this project we use a flow chart in order to understand by a simple human. After seeing the flow chart a non technological man can understand that how to work this calculator.**

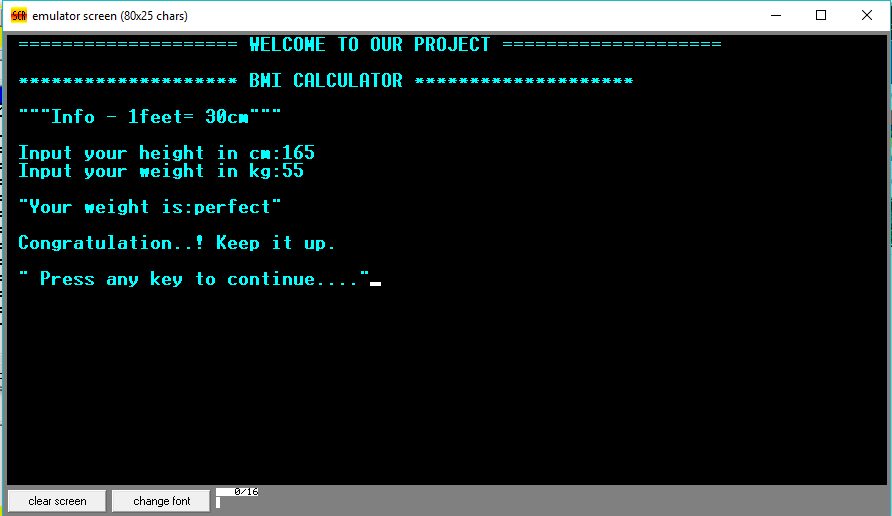
**Here first we start from START. Then we gave 2 input one for height and one for weight.**

**Then we finally can see the result after BMI calculation.**

**Experimental Result:**

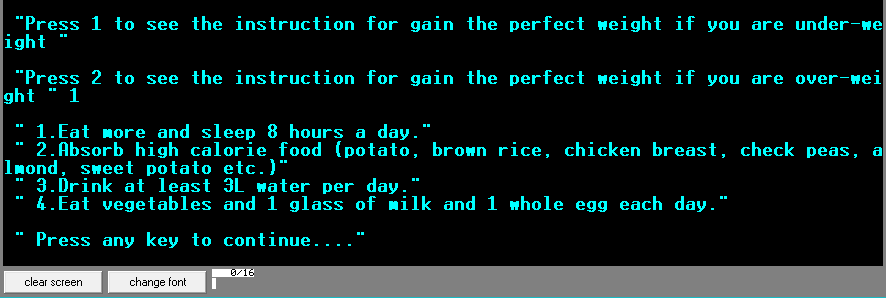
**In this project we use many things. Emu 8086 like**

* **Ascii code**
* **Loop, jump, cmp, AND etc..**

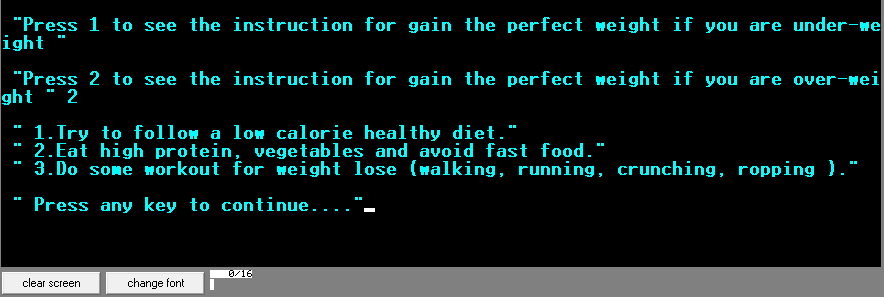
****

**Here is a short example for perfect weight.**

**Next we can see that there have 2 instructions. After pressing 1 we can see that how can we gain our perfect weight if we are in under weight.**

****

**If we press 2 then we can see that how can we gain our perfect weight if we are in over weight.**

****

**Advantage:**

* **Easy to calculate.**
* **Easy to Understand.**
* **Inspired to learn more**
* **Inspired to know more**
* **We can find our mass index and take proper steps for keeping fit our body at any time.**

**Disadvantage:**

* **We can't find actual BMI.**
* **This process can't take floating number and can't gave actual weight.**

**Future Work:**

**The BMI calculator provides innumerable opportunities for further investigation into the evolution of a task prioritization sceme within a dynamically changing ,randomly updated environment**

**Conclusion:**

**This project is interesting and helpful. The BMI calculator is inspired to create other calculator. We have completed our project and obtain more experience.**