Differences between Procedures and Macros

|  |  |  |
| --- | --- | --- |
| **Characteristic** | **Procedure** | **Macro** |
| **Number of Instructions that can be effectively handled by the microprocessor** | It is better to use Procedures for a set of a large number of instructions. Hence, it is optimal to use Procedures when the number of instructions is more than 10. | Macros are useful over Procedures when the number of instructions in the set is less. Therefore, when the subprogram contains less than 10 instructions, Macros are more efficient to use in such cases. |
| **Assembler Directives used** | The assembler directive - PROC is used to define a Procedure. And the assembler directive - ENDP is used to indicate that the body of the procedure has ended. | The assembler directive- MACRO is used to define a Macro, And to indicate that the body of the procedure has ended, the assembler directive- ENDM is used. |
| **Execution Process** | Every time a procedure is called, [the CALL and RET instructions](https://www.includehelp.com/embedded-system/the-call-and-ret-instruction-in-the-8086-microprocessor.aspx) are required for shifting the control of instruction execution. | Every time a Macro is called, the assembler of the microprocessor places the entire set of instructions of the Macros in the mainline program form where the call to the macro is made. |
| **Execution Time** | The Procedures execute slower than the Macros because every time a procedure is called, it is necessary to integrate and link it with the calling program and this takes time. | The execution of macros is faster as compared to procedures because there is no need to integrate or link the macros with the calling program. It is simply loaded into the main memory every time it is called. |
| **Overhead time** | Overhead time occurs while calling the procedure and returning the control to the calling program. | Overhead time is avoided as calling and returning does not take place. |
| **Amount of memory required** | The Procedures require less amount of memory than the Macros because a Procedure is written and loaded into the main memory only once, and is linked to the calling program when called. | The Macros require a large amount of memory because it is loaded into the main memory every time it is called. |
| **Number of times machine code generated** | The machine code (containing the instructions within the Procedure) is generated only once when the procedure is defined. | The machine code (containing the instructions within the Macros) is generated every time the macro is called. |
| **Passing of parameters** | In procedures, we cannot pass the parameter to id directly. However, the values can be passed to it using registers and also via stack. | The macros are capable of handling parameters within their definition and we can pass them in the statement which calls the macro. |