A micro assembler for a processor

INTRODUCTION

A micro assembler (sometimes called a meta-assembler) is acomputer programthat helps prepare amicroprogramto control the low level operation of a computer in much the same way anassemblerhelps prepare higher level code for aprocessor. The difference is that the microprogram is usually only developed by the processor manufacturer and works intimately with the hardware. The microprogram defines theinstruction setany normal program (including bothapplication programsandoperating systems) is written in. The use of a microprogram allows the manufacturer to fix certain mistakes, including working aroundhardwaredesign errors, without modifying the hardware. Another means of employing micro assembler-generated micro programs is in allowing the same hardware to run differentinstruction sets. After it is assembled, the microprogram is then loaded to astore to become part of the logic of aCPU'scontrol unit.

Some micro assemblers are more generalized and are not targeted at single computer architecture. For example, through the use of macro-assembler-like capabilities, Digital Equipment Corporationused theirMICRO2micro assembler for a very wide range of computer architectures and implementations.

If a given computer implementation supports awriteable control store, the micro assembler is usually provided to customers as a means of writing customized microcode.

è Computer programs(softwareprograms) areinstructionsfor acomputer. A computer requires programs to function, typicallyexecutingthe program's instructions in acentral processor. The program has anexecutableform that the computer can use directly to execute the instructions. The same program in its human-readable sourceform, from whichexecutableprograms are derived (e. g., compiled), enables aprogrammerto study and develop itsalgorithms.

Computer source code is often written by professionalcomputer programmers. Source code is written in aprogramming languagethat usually follows one of two mainparadigms: imperativeordeclarativeprogramming. Source code may be converted into anexecutable file(sometimes called an executable program or a binary) by acompiler. Alternatively, computer programs may be executed by acentral processing unitwith the aid of aninterpreter, or may beembeddeddirectly intohardware.

ASSEMBLY LANGUAGE:

A program written in assembly language it basically contain of a series ofinstructions- mnemonics that correspond to a stream of executable instructions, when translated by anassembler that can be loaded into memory and executed.

For example, anx86/IA-32processor execute the below instruction as given inmachine language.

Binary: 10110000 01100001 (Hexadecimal: B0 61)

The mnemonic “ move” it tells the opcode1011tomovesthe value in the 2nd operand into the register.

Transforming the assembly language into the machine code is done by anassembler, and the vice versa by this assembler. In High-level language, there is usually aone to one relationbetween simple assembly logics and machine language instructions. But, in few cases, an assembler provides instructionswhich will expand into several machine code instructions to provide its functionality. Eachcomputer structureandprocessor designhas its own machine understanding language. Each instruction is simple enough to be executed using a relatively small number of electronic circuits. System may differ by the type of operations they support. For example, a new 64-bit (AMD processor) machine will have different structure from a 32-bit (Intel processor) machine. They also have different size structure and the different storage structure. Multiple collection ofmnemonics codesor assembly-language code 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.

ASSEMBLER

The advancedassemblercreatesobject codeby translating assembly instruction mnemonics intoop codes, and by resolvingsymbolic namesfor memory locations and other entities. The use of symbolic references is only the key feature of assemblers, saving tuff calculations and manual address. Most assemblers also includemacrofacilities for performing textual substitution.

E. g.: To generate common short sequences of instructions to runinline, instead of in asubroutine.

Assemblers are easier to write thancompilersforHLL. Advanced assemblers, like RISCbased architectures, such asMIPS, SunSPARC, and HPPA-RISC, it optimizeinstruction schedulingto exploit theCPU pipelineefficiently.

There are two types of assemblers, based upon how many passes through the source are needed to produce the executable program. One-pass assemblers go through the source code once at a time and assume that all symbols will be defined before any instruction that references them. Two-pass assemblers create a table with all unresolved symbols in the first pass, and then use the 2nd pass to resolve these addresses.

The advantage in the one-pass assemblers is about its speed, which is not as important as it once was with advances in computer speed and capabilities. The advantage of the two-pass assembler is that symbols can be defined anywhere in the program source so it’s an easier way to understand the user. Its results to the program can be defined in a more logical and meaningful way. It will make two-pass assembler programs easier to read and maintain.

More sophisticatedhigh-level assemblersprovide language abstractions such as:

- Advanced control structures.
- High-level procedure declarations and invocations.
- High-level abstract data types, including structures, unions, classes, and sets.
- Sophisticated macro processing.
- Object-Orientedfeatures such asencapsulation, polymorphism, inheritance, interfaces.

Here's how it works:

- Most computers come with a specified set of very basic instructions that correspond to the basic machine operations that the computer can perform. For example, a " Load" instruction causes the processor to move a string of bits from a location in the processor'smemoryto a special holding place called aregister.
- The programmer can write a program using a sequence of these assembler instructions.
- This sequence of assembler instructions, known as thesource codeor source program, is then specified to the assembler program when that program is started.
- The assembler program takes each program statement in the source program and generates a corresponding bit stream or pattern.
- The output of the assembler program is called theobject codeor object program relative to the input source program.
- The object program can then be run whenever desired.

Earlier programmers actually wrote programs in machine code, but assembler languages or instruction sets were soon developed to speed up programming field. Today, assembler programming is used only where very efficient control over processor operations is needed. It requires knowledge of a particular computer's instruction set. Historically, most programs have been written in " higher-level" languages such as COBOL, FORTRAN, PL/I, and C. These languages are easier to learn and faster to write programs with than assembler language.

MICROASSEMBLER

A micro assembler also called as meta-assembler. It is a kind of program which helps prepare a micro program to control the low level operation of a computer in much the same way an assembler helps prepare higher level code for a processor. The use of a micro program allows the manufacturer to fix certain mistakes, in hardware design also. Another means of employing micro assembler-generated micro programs is in allowing the same hardware to run different instruction sets. When it is assembled, the micro program is then loaded to a control store to become part of the logic of a CPU's control unit.

Some micro assemblers are more generalized and are not targeted at single computer architecture.

For example, the use of macro-assembler likes capabilities, Digital

Equipment Corporation used their MICRO2 micro assembler for a very wide range of computer architectures.

THE ASSEMBLER USED BY THE MICROSOFT (MASM)

MASM is a Microsoft's assembler and abbreviation used for it is " Macro Assembler." MASM is a very powerful macro feature, and is capable of writing very low-level syntax, and pseudo-high-level code with its macro feature. MASM 6. 15 is currently available as a free-download from Microsoft site.

MASM is a one of the Microsoft development tools that are targeted 16-bit, 32-bit and 64-bit platforms. Versions 6. 1 and 6. 11 included Phar Lap's TNT DOS extender so that MASM could run in MS-DOS.

- MASM will write in Intel Syntax.
- MASM is used by Microsoft to implement some low-level portions of its Windows Operating systems.
- MASM, contrary to popular belief, has been in constant development since 1980, and is upgraded on a needs-basis.
- MASM has always been made compatible by Microsoft to the current platform, and executable file types.
- MASM currently supports all Intel instruction sets, including SSE2.

MAL (MICRO ASSEMBLY LANGUAGE):

It describes about the lexical, syntactic, and semantic elements of the language, and gives a focus on microprogramming with the mic1 micro-assembler.

Lexical:

Most assembly language such as Micro-Assembly Language is a line-oriented language. Each micro-instruction is generally defined on a single line of the program file. The end-of-line is generally significant. It is a case-sensitive. For example, “ AND” is a reserved word

Corresponding to a bitwise operation of the mic1 ALU, while “ and” is not reserved and may be used as a label

Comments

The comments will begin with two slash characters (“//”) and continue to the end of the line. Blank lines and lines consisting only of white space and comments are ignored by the micro-assembler.

Directive

Directives for the micro-assembler begin with a period character (“.”) and may contain alphabetic characters.

There are two micro-assembler directives: “. default” and “. label”. Directives are used to provide guide the behavior of the micro-assembler, and do not correspond with words in the control store.

Reserved Words

The names of registers and control lines are reserved, as are the words “ if”, “ else”, “ goto”, “ nop”, “ AND”, “ OR”, and “ NOT”. For the mic1 architecture, the following words are reserved and may not be used as statement labels:

MAR

MDR

PC

Fetch

If

Else

goto

nop

AND

OR

NOT

MORE ABOUT THE MICRO ASSEMBLER:

Micro Assembler is Integrated Development Environment for assembly programming.

Micro Assembler has a much easier syntax than any of the major assemblers, a great combination for beginners.

Micro Assembler is a Windows based application so you can enjoy user-friendly Windows environment.

APPLICATIONS

Hard-coded assembly language is typically used in a system'sboot ROM(BIOSon IBM-compatiblePCsystems). This low-level code is used, among other things, to initialize and test the system hardware prior to booting the OS, and is stored inROM. Once a certain level of hardware initialization has taken place, execution transfers to other code, typically written in higher level languages; but the code running immediately after power is applied is usually written in assembly language. The same is true of mostboot loaders.

Many compilers render high-level languages into assembly first before fully compiling, allowing the assembly code to be viewed fordebuggingand optimization purposes. Relatively low-level languages, such asC, often provide specialsyntaxto embed assembly language directly in the source code. Programs using such facilities, such as theLinux kernel, can then construct abstractions utilizing different assembly language on each hardware platform. The system’sportablecode can then utilize these processor-specific components through a uniform interface.

Assembly language is also valuable inreverse engineering, since many programs are distributed only in machine code form, and machine code is usually easy to translate into assembly language and carefully examine in this form, but very difficult to translate into a higher-level language. Tools such as theInteractive Disassemblermake extensive use of disassembly for such a purpose.

A particular niche that makes use of assembly language is thedemo scene. Certain competitions require the contestants to restrict their creations to a very small size (e. g. 256B, 1KB, 4KB or 64 KB), and assembly language is the language of choice to achieve this goal. When resources, particularly CPU-processing constrained systems, like the earlierAmiga models, and theCommodore 64, are a concern, assembler coding is a must: optimized assembler code is written “ by hand” and instructions are sequenced manually by thecodersin an attempt to minimize the number of CPU cycles used; the CPU constraints are so great that every CPU cycle counts. However, using such techniques has enabled systems like the Commodore 64 to produce real-time3D graphicswith advanced effects, a feat which might be considered unlikely or even impossible for a system with a 0. 99MHzprocessor

BENEFITS OF IT:

The micro programmed Data General Eclipse S/200 computer is available with a writable control store. The WCS feature of the Eclipse is having extension of the micro programmed control logic of the computer's central processing unit. It allows a user to implement specialized instructions at a very low level. Its use is however, discouraged since Data General does not provide software support for the WCS feature.

BIBLIOGRAPHY:

1. Microprogramming with the Eclipse Computer WCS feature Corporation, 1974.
2. www. wikipedia. com
3. www. google. com
4. www. ontko. com
5. Answers. com. " assembly language: Definition and Much More from Answers. com". Retrieved 2008-06-19.
6. NESHLA: The High Level, Open Source, 6502 Assembler for the Nintendo Entertainment System
7. Eidolon's Inn : SegaBase Saturn