VI. Processor

Processor is where sequence of binary codes are translated to do certain functions. These codes are broken into many parts (usually op code, register source, register destination, function code for general purpose register ISA) and converted into control signals and variables for proper functions.

VII. Run-time Environment

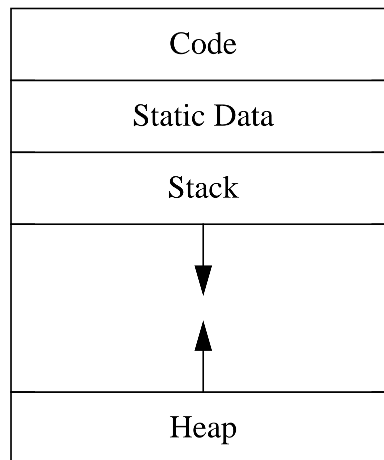

Fig. 2 Typical subdivision of run-time memory into code and data areas.

This section is a brief description of higher level of processing in order to describe the conceptual workings of a processor.

When code is executed by a processor, parts of program (stack and heap) will grow and shrink. In execution of code sometimes use of stack is necessary for efficient use of resource. Instead of using random memory location for new run-time environment, allocating space using stack is much efficient. This is because executing process can be pushed into available memory and immediately popped out once its function is served. Common way to achieve this is by using what is call an activation record, which is maintained by two dedicated registers, stack pointer and frame pointer.

VIII. Closing Remarks

Concept discussed above is a short description of processing steps that range from compiler down to processor. To give a high-level perspective on the concept, a brief description of the run-time environment is also described.

Reference

1. Aho, A., Sethi, R., Ullman, J. **Compilers: Principles, Techniques, and Tools.** Dalton Publisher, 1988.

II. Preprocessor

Preprocessors produce input to compilers. They may perform many different processes before passing the source code to compiler:

1 All the user defined macros maybe processed and replaced by the longhand codes.
2 Preprocessor also may include all the header files into the program text.
3 Some preprocessors called “Rational” preprocessor may augment older languages with more modern flow-of-control and data-structuring facilities.
4 There are also some language extension by adding the built-in macros in the program. These are sometimes added as comments with C compiler ignores but the preprocessor would recognize.

III. Compiler

A compiler is a program that reads a program written in one language and translate it into an equivalent program in another language. Thus, after preprocessing, the source code is passed onto compiler as a complete program.

A compiler then goes through many steps to verify the lexical and semantic aspects of the code. Basic phases of the compiler are 1. lexical analysis, 2. syntax analysis, 3. semantic analysis, 4. intermediate code generation, 5. code optimization, and 6. code generation. After these phases, a new code equivalent in function is generated.

Compilers, thus, does not have to produce assembly code for a loader/linker. Some compilers can be sequentially linked to produce different codes to produce acceptable assembly code for a particular machine. This feature of compiler makes code relatively easy to port from one architecture to another.

IV. Assembler

Some compilers produce assembly code which is then passed to another assembler for more processing. Others may produce a relocatable machine code that is passed to loader/linker editor. Assembly language is a direct english translation of machine code. These codes do not have specific addresses assigned to them, so they can be relocated and augmented by linker. This makes program flexible and easy to control. Then relocatable code is translated into machine code which is made of binary coding.

V. Loader and Link-Editor

Loader does the job of determining the exact addresses to place many segments of code. This process is called linking. By linking and placing the relocatable machine code, absolute machine code can be made. Absolute machine code is then loaded into memory for execution.

University of California, Berkeley
Computer Science Division
Department of Electrical Engineering and Computer Sciences
Berkeley, Ca 94720

Spring 1996 Y. Cho

Supplementary Notes: Language to Code

CS 152: Discussion 102

I. The Big Picture

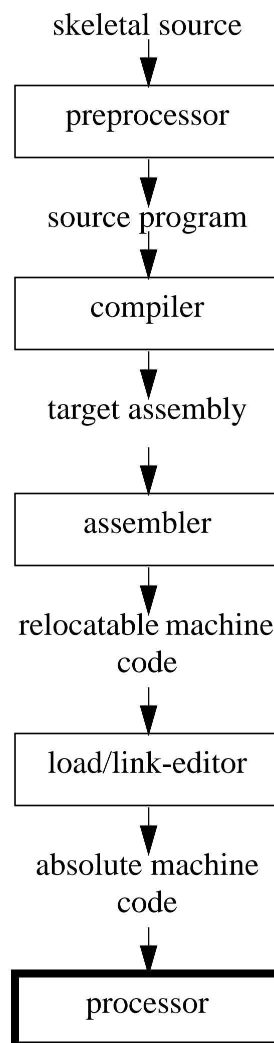

Fig. 1 A language processing system