6.1.1 プログラミング言語によるソフトウェア作成

MPUが理解する機械語と人が作成する高級言語

組込み機器で実行されるプログラムは，機械語でできている．機械語に対応する言語がアセンブリ言語だ．C言語などの人間が理解しやすい高級言語でプログラミングし，これをコンパイルして機械語を生成するのが一般的だ．

Key Word

機械語(マシン語)，アセンブリ言語，アセンブラ，アセンブル，C言語，コンパイラ，コンパイル，オブジェクトコード，高級言語(高水準言語)，低級言語(低水準言語)，プログラム，プログラミング言語，プログラマ，ソースコード，リンク，ロードモジュール，リンカ，ライブラリ，静的リンク，動的リンク

組込み機器にはMPUが実装されており，これにソフトウェアが指示を与えることで意味のある動作をする．MPUが理解できるのは，電圧がHighかLowかだけであり，その組み合わせをさまざまな指令として受け取る．ソフトウェアは，このHighとLowを意味のある組み合わせや順序でMPUに指令を与える．このソフトウェアの中ではHighとLowを数値の1と0(またはその逆)で表現し，その組み合わせや順序で，データや命令が記述される．このデータや命令を**機械語**(**マシン語**)と呼ぶ．

マシン語は人間にとって理解するのが難しいため，これをある程度容易にするために，機械語を意味のある略語で表現したものが**アセンブリ言語**(**Assembly Language**)である．アセンブリ言語はそのままでは実行できず，**アセンブラ**(**Assembler**)というプログラムによって機械語に変換する(**アセンブル**，**Assemble**)必要がある．アセンブリ言語は機械語に対応しており，MPUに依存する．

アセンブリ言語より，さらに人間に理解しやすく表現されたものが**C言語**で，組込みシステムでよく使われる．C言語もそのままでは実行できず，**コンパイラ**(**Compiler**)というプログラムによって機械語に変換する(**コンパイル**，**Compile**)必要がある．C言語は移植性が良く，MPUに対応したコンパイラを利用することで，さまざまなMPUに対応したプログラムを生成できる．コンパイラによって生成された機械語は，それ単体では実行できない中間的なもので，**オブジェクトコード**(**Object Code**)と呼ばれる．

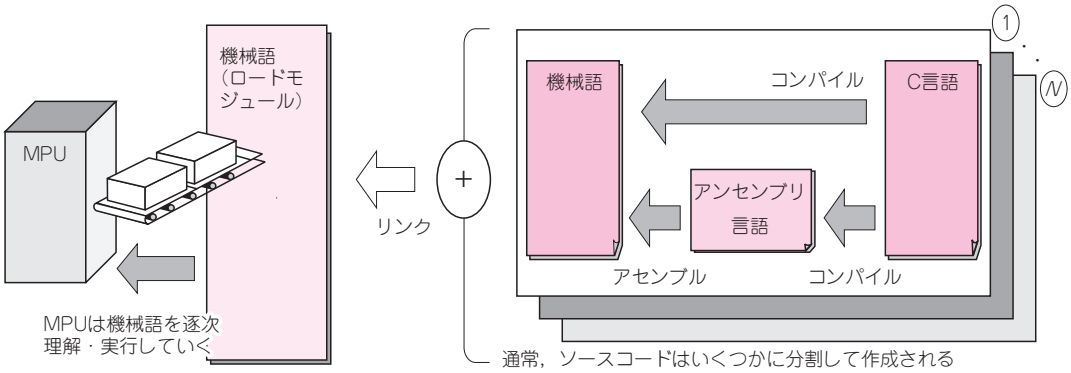
OSやデバイスドライバなどのMPUや周辺機器を直接制御したい場面では，C言語のコンパイラに依存せず，直接アセンブリ言語で記述することが多い．

C言語のように，可読性が高い言語を**高級言語**(または**高水準言語**)と呼び，アセンブリ言語のように機械語に近い言語を，機械語も含んで**低級言語**(または**低水準言語**)と呼ぶ．

また，MPUの処理の手順を示すものを**プログラム**(**Program**)といい，高級言語や低級言語に関わらず，これを記述するための言語を**プログラミング言語**(**Programming Language**)という．プログラムを作成する人を**プログラマ**(**Programer**)という．

ソースコード(**Source Code**，C言語などで記述された一連の命令など)は，エディタ(Editor)などを

プログラムが生成され実行されるまで



用いて通常いくつかに分割して作成される．分割されたソースコードはコンパイルによりオブジェクトコードが生成され，さらにこれを結合（**リンク**）することで，MPUが読み取って実行する**ロードモジュール**ができあがる．リンクするプログラムを**リンカ**（**Linker**）と呼ぶ．

場合によっては，汎用的に利用できるプログラムが用意されていることがある．これを**ライブラリ**（**library**）と呼ぶ．ソフトウェアのすべてを自前で作成することもあるが，ライブラリを使って一部の機能を実現することが多い．ライブラリは，オブジェクトコードで提供される場合やソースコードで提供される場合がある．

リンクには静的リンクと動的リンクの2種類があり，前者はリンク時にすべてのオブジェクトコードやライブラリを一つのモジュールに結合するもので，後者はプログラム実行時に必要に応じてライブラリを取り込むものである．組込みシステムでは，静的リンクが採用されることが多い．

要点のチェック

□ MPUが理解できるデータや命令を**機械語**（**マシン語**）といい，**1**と**0**の意味のある組み合わせや順序でできている．これを意味のある略語で表現したものがアセンブリ言語で，**アセンブラ**というプログラムを使い**機械語**に変換（アセンブル）する．

□ 人間にとって可読性が高い言語を**高級言語**と呼び，MPUが理解できるデータや命令に近い言語を**低級言語**と呼ぶ．C言語は**高級言語**の一つであり，組込みシステムでよく使われる．C言語で書かれたプログラムは，**Cコンパイラ**というプログラムによって，機械語に変換（コンパイル）される．MPUに対応した**コンパイラ**を利用することで，さまざまなMPUに対応したプログラムを生成できるなど，**移植性**が良い．コンパイラによって生成された機械語は，それ単体では実行できない中間的なもので**オブジェクトコード**と呼ばれる．

□MPUの処理の手順を示すものを**プログラム**といい，これを記述するための言語を**プログラミング言語**という．また，これを作成する人を**プログラマ**という．

□ ソースコードは，通常いくつかに分割して作成される．これらはコンパイルされて機械語になったあと，**リンク**（これを実行するプログラムを**リンカ**という）されてロードモジュールになる．

□ ソフトウェアの一部の機能を，**ライブラリ**という汎用的に利用できるプログラムを使って実現することもある．このプログラムは，**オブジェクトコード**で提供される場合やソースコードで提供される場合がある．

6.1.2 C言語に関する規格や規約

C言語を規定する規格と開発時の約束(規約)

C言語の文法などは規格で規定されており，これにしたがって記述しなくてはならない．さらに，信頼性，保守性，移植性，効率性などを良くするため，企業や個別プロジェクトでコーディング規約を決めていることが多く，その遵守が求められる．

Key Word

C言語，カーニハンとリッチー，ISO，ANSI，C90，日本工業規格，C99，C++，Java，コーディング規約，信頼性，保守性，移植性，効率性，MISRA，MISRA-C

C言語(C Language)は，もともとAT&Tベル研究所で作られたプログラミング言語である．UNIX上で開発され，UNIX上で走るプログラムの大部分がC言語で書かれている．C言語はコンパイラやOSを書くのにも使えることからシステム・プログラミング言語と呼ばれている．C言語の定義は**カーニハンとリッチー**(BrianW.Kernighan&DennisM.Ritchie)により書かれた「The C Programming Language」などが事実上の標準とされていたが，あいまいな部分があったため**ISO**(**国際標準化機構**：International Organization for Standurdization)と**ANSI**(**アメリカ規格協会**：American National Standards Institule)が標準化を進め，1990年に「INTERNATIONAL STANDARD ISO/IEC 9899：1990(E)Programming Languages-C」(通称**C90**)が発行された．これを翻訳し，JIS X3010-1993 プログラム言語Cとして**日本工業規格**(**JIS**)となっている．その後，1999年にISOで規格改定され，「INTERNATIONAL STANDARD ISO/IEC 9899：1999(E)Programmin Languages-C(Second Edition)」(通称**C99**)が発行された．日本工業規格版は，「JIS X3010-2003 プログラミング言語C」である．C99では，C++の機能のいくつかが取り込まれている．

下表に代表的なプログラミング言語と，その規格などについてまとめる．

代表的なプログラミング言語

プログラミング言語	規格など
アセンブリ言語	各プラットフォームに対応した様々なものがあるが規格はない 情報処理技術者試験用として仮想計算機 COMET Ⅱ用の言語が CASL Ⅱとして定義されている
C 言語	ISO/IEC 9899：1999(E)Programmin Languages-C(Second Edition) (日本)JIS X3010-2003 プログラミング言語 C
C++	ISO/IEC 14882：2003　Programming languages -- C++ (日本)JIS X3014-2003 プログラミング言語 C++
Java	JCP(Java Community Process：Java プラットフォームの将来に関与する標準化プロセス)による

さらに，ソースコードの書き方自体を制限する**コーディング規約**がある．規格にしたがうだけでは，その書き方に自由度がありすぎ，不用意にバグなどを埋め込む可能性が高かったり，移植性の悪いコードとなってしまったりすることが多い．これを防ぐために，おもに四つの視点［信頼性(Reliability)，

保守性(Maintainability)，移植性(Portability)，効率性(Efficiency)］から自由度をある程度制限して個人的な品質のばらつきをなくすように，企業ごと，もしくはプロジェクトごとに独自にコーディング規約を設けることが多い．

コーディングに関わる品質特性

組込みソフトウェア開発向けコーディング作法ガイド［C 言語版］より	
信頼性	指定された条件下で利用するとき，指定された達成水準を維持するソフトウェア製品の能力
保守性	修正のしやすさに関するソフトウェア製品の能力
移植性	ある環境から他の環境に移すためのソフトウェア製品の能力
効率性	明示的な条件の下で、使用する資源の量に対比して適切な性能を提供するソフトウェア製品の能力

コーディング規約を遵守することで，具体的には，①不具合が少なく，エラーに対する許容能力が高くなり(信頼性)，②コードが理解・修正しやすく，修正の影響が小さくでき(保守性)，③異なる環境へ移植しやすくなり(移植性)，④時間や資源の利用効率が高くできるようにすること(効率性)をめざしている．

自動車の安全や信頼に関わるソフトウェア向けにヨーロッパの**MISRA**(**The Motor Industry Software Reliability Association**)によって定められた**MISRA-C**(**Guidelines for the Use of the C Language in Vehicle Based Software**)などがある．また，IPAからは，「組込みソフトウェア開発向けコーディング作法ガイド［C言語版］」というガイドラインが発行されている．

コーディング規約が守られているか否かを評価するために，静的解析ツールを利用すると便利である．MISRA-Cに準拠しているか否かを評価できるツールもある．

コーディング規約例

信頼性に関すること
- 条件の検討が漏れていないことが明示的になるように `if` 文では `else` 節を省略しない，`switch` 文では `default` 節を省略しない

保守性に関すること
- 変数のスコープや型がわかりやすいように，プリフィックスで区別できるようにする
- スコープがわかりやすいようにインデントを統一する
- 可読性を悪くしないため，多重ループ抜けなど以外に `goto` 文を使わない（複雑になれば信頼性にも関係する）
- 関数名はその機能を理解しやすくするように「動詞」もしくは「動詞＋目的語」の形で表現する

移植性に関すること
- コンパイル環境に依存しないように，ファイル参照先をパスで絶対指定しない

効率性に関すること
- スタックを無駄に消費しないように，大きなサイズの変数を引数で渡すときはポインタを使う

要点のチェック

□ C言語は，**カーニハン**と**リッチー**による著書「The C Programming Language」が事実上の標準として利用されていた．この標準化を**ISO**と**ANSI**が進め，1990年に「INTERNATIONAL STANDARD ISO/IEC 9899：1990(E)Programming Languages-C」(通称**C90**)が発行された．これを翻訳したものが**日本工業規格**(**JIS**)になっている．

□ その後，1999年にISOで規格改定され，「INTERNATIONAL STANDARD ISO/IEC 9899：1999(E)Programming Language-C(Second Edition)」(通称**C99**)が発行された．これに対する日本工業規格版は，「JIS X3010-2003 プログラミング言語C」である．C99では，**C++**の機能のいくつかが取り込まれている．

□ コーディング規約は，ソースコードの書き方を**制限**することでもある．コーディング規約を守ることで，**信頼**性，**保守**性，**移植**性，**効率**性などが向上する．