ホストマシン上とシミュレータ上の共存実行

6N-6

村上知嘉子　川北英幸　平尾繁晴
株式会社東芝　システム・ソフトウェア技術研究所

1．はじめに

ソフトウェア生産の工業化をめざすＩＭＡＰシステム（Integrated software MAnagement and Production support system）[1]の一環として、マイクロコンピュータソフトウェアのクロス開発環境の研究開発を行なっている。

クロスソフトウェア開発では、プログラムのコンパイル／アセンブル、シミュレータ上での論理テストまでをホストマシン上で行ない、論理テストの終わったプログラムを実機上で動作させてシステムテストを行なう。[2]クロスソフトウェア開発を行なうことにより、ホストマシンの持つプログラム開発のための豊富なリソースを用いることができる。また、実機のハードウェアが完成する前にプログラムテストを行なうことができ、より効率よくプログラムの開発を行える。

しかし、このシミュレータを用いた論理テストもシミュレータの実行速度が遅い場合は、大きなプログラムを実行するのに時間がかかり、実質的にはプログラム開発に効果があるとはいえなくなる。このような実行時間によるプログラム開発の制限を解決するためには、論理テスト時のプログラムの実行時間を短縮する必要がある。

本発表では、プログラムの実行時間を短縮するために、ホストマシン上で直接動作するモジュールとシミュレータ上で動作するモジュールをひとつのプログラムの中に共存させて実行することが可能な環境（共存実行環境）上でプログラムの実行を行なうという方法について検討を行なったので報告する。

2．共存実行環境の必要性

最近では、マイクロプロセッサの応用プログラム開発にＣ言語のような高級言語が使用されることが多くなってきた。Ｃ言語を用いてプログラムを開発する場合、ホストマシン用のＣコンパイラで再コンパイルすることにより、ホストマシン上で実行することが可能となり、プログラムを高速に実行し、動作を確認することができる。

しかし、実機用のアセンブリ言語を用いて記述されているモジュールやデバッグを行ないたいモジュールはシミュレータを用いて実行しなければならない。そのためプログラム中にシミュレータ上で実行したいモジュールがひとつしかない場合にも、結果としてプログラム中のすべてのモジュールをシミュレータ上で実行しなければならなくなる。

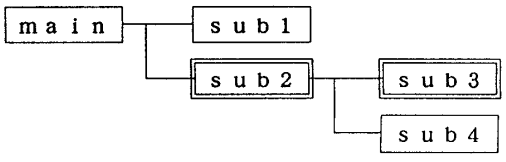
また、シミュレータは実機のプログラムを１命令ずつシミュレーションするため、プログラムの実行に時間がかかる。このことから、シミュレータ上で動作する命令数を少なくすれば、それに比例してプログラムの実行時間は短くなる。

そこで、ひとつのプログラムの中で、シミュレータ上で実行するモジュールとＣ言語で記述されたホストマシン上で実行するモジュールを共存させて実行する環境を考えた。この共存実行環境でプログラムを実行することで、シミュレータを用いて実行する命令数を少なくすることができ、プログラムの実行時間を短縮することが可能になると考えられる。

3．共存実行環境の作成

ここで、図．１のような構成を持つプログラムを実行する場合について考えることにする。

このプログラムは五つのモジュールmain、sub1、sub2、sub3、sub4から構成されている。モジュールmain、sub1、sub4の三つは、Ｃ言語を用いて記述されており、すでにテスト済みでホストマシン上でも動作することが確認されているものとする。



図．１　プログラムの構成図

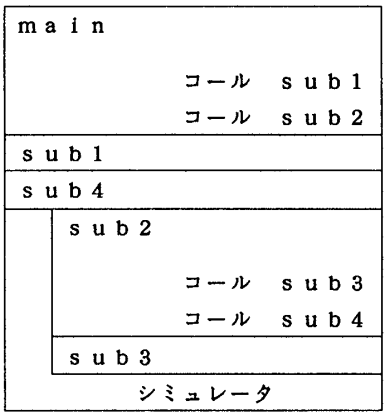
いま、モジュールmain、sub1、sub4はホストマシン上で直接実行し、モジュールsub2とsub3の二つのモジュー

Coexistence execution on a hostmachine and a simulator
Chikako Murakami, Hideyuki Kawakita, Shigeharu Hirao
TOSHIBA Corporation Systems & Software Engineering Lab.

ルはシミュレータ上で実行する。

このプログラムを共存実行環境に置いたときの構成をを図．２に示す。

このとき、モジュールsub2を呼ぶ命令はシミュレータ上のモジュールsub2を呼ぶ命令に変更する。この命令は、sub2を実行開始アドレスとしてシミュレーションを開始することで実現している。同様にモジュールsub2内でモジュールsub4を呼ぶ命令は、ホストマシン上のモジュールsub4を呼ぶ命令にあらかじめ変更しておく。ホストマシン上のモジュールはシミュレータ上のモジュールから直接呼び出すことができないので、一旦制御をシミュレータに移した後、呼び出される。また、シミュレータはホストマシン上でモジュールsub4の実行が終了した後、シミュレーションを再開するために現在のプログラムカウンタの値を覚えておく。



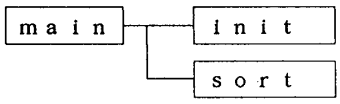
図．２　共存実行が可能な環境の構成図

この図．２にそってプログラムの実行方法を示す。

プログラムはモジュールmainから開始する。モジュールsub1を呼ぶとモジュールsub1はホストマシン上のモジュールであるため、通常通り実行する。モジュールsub2を呼ぶときには、あらかじめ命令を変更しておいたようにシミュレータ上のモジュールsub2を呼び、シミュレータ上で実行する。モジュールsub3はシミュレータ上のモジュールなので、通常のシミュレーションを行なう。モジュールsub4はホストマシン上のモジュールなので、呼び出すときには事前に変更しておいた命令を実行し、一度シミュレーションを中断して、ホストマシン上で実行する。

４．共存実行環境を用いた場合の効果

この共存実行環境を用いて、図．３のような構成を持つソートを行なうプログラムの実行時間を調べた。



図．３　ソートを行なうプログラムの構成図

このプログラムのすべてのモジュールをシミュレータ上で実行する場合の実行時間を基準にして、プログラムの一部のモジュールをホストマシン上で実行し他の残りモジュールはシミュレータ上で実行する場合の時間の比較を表．１に示す。

main	init	sort	時　間
■			０．６
■	■		３１．７
■		■	６９．８
■	■	■	１００．０

□ ホストマシン上
■ シミュレータ上

表．１　実行時間の比較

プログラムの全モジュールをシミュレータ上で実行したときの実行時間を基準とすると、モジュールmainのみをシミュレータ上で実行した場合は１％以下、モジュールmainとモジュールinitをシミュレータ上で実行した場合は約３０％、モジュールmainとモジュールsortをシミュレータ上で実行した場合は約７０％、にそれぞれプログラムの実行時間を短縮できることが表．１から確認できる。

５．結論

今回検討を行なったホストマシン上のモジュールとシミュレータ上のモジュールをひとつのプログラムの中に存在させて実行することが可能な共存実行環境を用いることにより、論理テスト時のプログラムの実行時間を短縮できることが確認できた。

参考文献

［１］大筆　他：”ＩＭＡＰシステム（１）－概要－”，情報処理学会第３１回全国大会，1985,9

［２］井上　他：マイコンソフト開発における論理テスト環境、情報処理学会第３５回全国大会、1987,9