COBOL マシンとその設計思想†

——ハードウェア構成について——

山本昌弘†† 中崎良成††
横田 実†† 梅村 護††

本論文は COBOL ステートメントとほぼ一対一に対応する命令などを備えた高度なアーキテクチャを持つ COBOL マシンのハードウェア構成と設計思想について論じている．COBOL マシンは高度なアーキテクチャを高性能に実現するために，3つのプロセッサから成る複合体で構成され，各マシン命令はパイプライン方式により並列に実行される．そして，各プロセッサはそこで必要な処理を高速に行うために，ファームウェアとハードウェアによって専用化されている．

本 COBOL マシンは製作を完了し，汎用計算機 ACOS システム 300 をもとに COBOL マシンとの付加機構などを組み込まれて実現された計算機に接続されて動作している．そして，各種の応用プログラムを COBOL マシン上で実行することによって，本 COBOL マシンと同レベルのハードウェア技術の汎用計算機よりも高速に実行されることが明らかになった．

1. はじめに

著者らは COBOL プログラムを高速に実行する構造を備えた COBOL マシンを開発したが，本論文では COBOL マシンのハードウェア構成について論じている．

これまでに，COBOL マシンの開発は行われているが[1),2)]，これらは，COBOL の言語仕様が低く（ANSI 74 仕様[3)]を満たしていない），また，中間言語仕様やハードウェア構成の機能レベルも低い．このために，高い機能，性能が達成されていないと考えられる．

本 COBOL マシンは ANSI 74 COBOL 言語仕様を満たし，COBOL の言語機能に直接対応する高い中間言語仕様を持ち，かつ，中間言語を効率良く処理する高度なハードウェア構成を備えており，これまでの COBOL マシンと比べて，アーキテクチャ，ハードウェア構成の両面から高級言語指向度が高い COBOL マシンを実現している．

高機能な COBOL マシンのアーキテクチャを高性能に実現するために，本 COBOL マシンは3つのプロセッサから成る複合体で構成される．そして，各プロセッサはそこで要求される処理を高速に行うために，ファームウェアとハードウェアによって専用化されている．また，COBOL マシンは汎用計算機に接続されて動作し，入出力処理などの実行を汎用計算機に依頼するための高性能なインタフェースを備えている．

なお，COBOL マシンのアーキテクチャについては文献 4) を参照のこと．

2. 設計方針

COBOL マシンのハードウェアの設計に当り，次に示す点を考慮して行った．

1）高性能の実現

アーキテクチャのレベルが高いと，1つの命令の実行に必要な処理が多くなる．これらの処理を高速に行うために，各命令を数個のフェーズに分け，それらを並列にパイプライン形式で実行する方式を採用する．COBOL マシンの命令は汎用計算機の命令より複雑なため，3つのフェーズに分けて（大型汎用計算機には命令フェッチと命令実行の2つのフェーズに分けて実行するものがある）実行する．そして，各フェーズに対応する処理を行うプロセッサはそこで必要な処理に専用化された構造を持つことによって，高速化を達成する．さらに，汎用計算機のように機械語命令へ展開されてしまうと元のソースステートメントとの対応がつかないが，本 COBOL マシンでは対応づけが可能である．これを積極的に利用して，COBOL マシン

† COBOL Machine and its Design Concept —Hardware Configuration— by MASAHIRO YAMAMOTO, RYOSEI NAKAZAKI, MINORU YOKOTA and MAMORU UMEMURA (C & C Systems Research Labolatories, Nippon Electric Co., Ltd.).

†† 日本電気(株) C & C システム研究所

はソースステートメントの動作を意識した処理を行うことにより，高速化をはかる．

2）LSI や VLSI 指向の構成

複数個のプロセッサに分割する際，1つのプロセッサが将来 LSI 化できる程度の大きさになるように行う．また，各プロセッサを構成する機能モジュールがプロセッサ間で共通に使用できるように配慮した．さらに，各プロセッサはビットまたはバイトスライス分割を行い，分割された処理モジュールがくり返し使用されるようにする．さらに，ROM, PLA, レジスタファイルなどのメモリ構造型素子の積極的利用，マイクロプロセッサをベースにした構成などを行う．

これらを実施することによって，LSI, VLSI 指向とともに，低価格化も達成することができる．

また，上記 1),2) のほかに，ホストプロセッサとの交信を高性能に行うための機構を実現するファームウェアおよびハードウェアを組み込むこととした．

3. プロセッサ構成

図1 は COBOL マシンの構成を示す．COBOL マシンが処理する命令は高度で複雑なため，3つのプロセッサ（命令取り出し，オペランド取り出しおよび命令実行プロセッサ）で構成される[5]．そして，これらをサポートするために，メモリインタフェース制御部，先取り制御部を備えている．

3つのプロセッサはパイプライン形式により並列処理を行っている．並列処理は汎用計算機でも行われている命令間のもの（**図2**(a) にその例を示す）と，多数オペランド命令の場合はオペランド間のもの（図2(b) にその例を示す）がある．図2(a) での 1,2,3,4 はそれぞれ命令を示す．そして，命令取り出しプロセッサ，オペランド取り出しプロセッサおよび命令実行プロセッサは各命令に関する命令取り出し，オペランド取り出しおよび演算処理を行う．一方，図2(b) は1つの COBOL マシンの命令 MOVE *A* TO *B C D* の実行過程を示す．そして，図で，命令取り出しプロセッサの *A, B, C, D* は命令列取り出し後のオペランドアドレスの展開処理，オペランド取り出しプロセッサの *A* : *B*, *A* : *C*, *A* : *D* はオペランドフェッチとデータタイプ変換，命令実行プロセッサの *A*→*B*, *A*→*C*, *A*→*D* は移送処理を示す．

このようなプロセッサの機能分割に当り，次の条件を満たすことを目標に行った．

a．各プロセッサの負荷をバランスさせる．

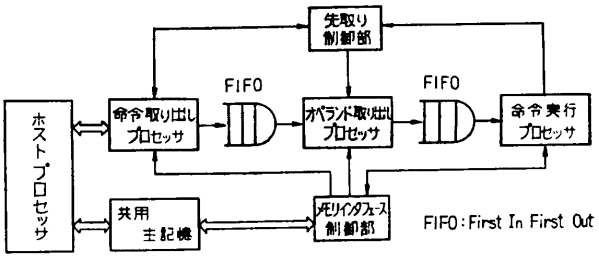
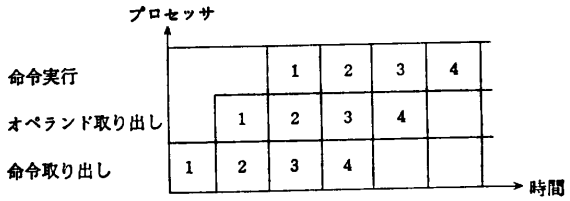
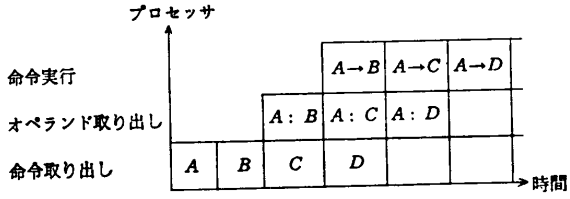


図 1 COBOL マシンの構成
Fig. 1 COBOL machine configuration.



（a） 命令間の並列処理
（図の 1,2,3,4 はそれぞれ命令を示す）



（b） オペランド間の並列処理
（1つの COBOL マシンの命令 MOVE *A* TO *B C D* の処理過程を示す）

図 2 COBOL マシンの並列処理
Fig. 2 COBOL machine parallel processing

b．プロセッサ間のインタフェースを単純にする．

c．各プロセッサで必要なハードウェアが重複しないようにする．

上記要求事項のうち，b と c は論理的な分析によって達成可能である．しかし，要求 a については，命令の種類やオペランドの属性によって各プロセッサの処理量が異なり，したがって，すべての場合について満たすことは困難である．このために，プロセッサ間を複数語の容量を持つ FIFO キュメモリを介してデータの授受を行うことによって，プロセッサでの処理量がバランスしないことによって生じるプロセッサ間の待ち合わせをできる限り小さくなるようにした．

4. 命令取り出しプロセッサ

本プロセッサは主に次に述べる処理を行う．

1）命令列のプリフェッチとソースステートメント

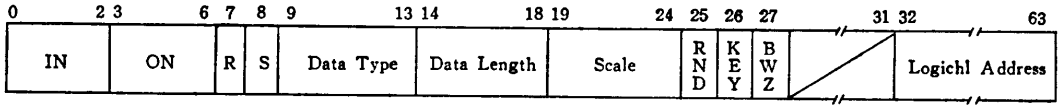


図 3 内部ディスクリプタの形式（算術データの場合）
Fig. 3 Internal descriptor format.

の動作を反映する高度な先取り処理

12バイト長の先取りバッファを備え，主記憶とのインタフェース幅である4バイト以上の空間ができると4バイト単位で命令列の先取りを開始する．そして，命令取り出しプロセッサの処理に必要な命令ストリームが常に存在するように制御する．

COBOL マシンの命令は翻訳前のソースステートメントとの対応が可能なように設定されている．したがって，これを利用することによりソースステートメントの動作を反映する高度な先取り処理が可能である[6]．

- GO TO や CASE 命令はこのプロセッサが分岐アドレスを決定し，その方向への命令取り出しを行う．
- IF 命令では条件が成立しない方向へ先取りする．
- PERFORM 命令では指定されたセクションまたはパラグラフに制御が移るように先取りを行う．そして，そのセクションまたはパラグラフの終端を検出し，元へ戻るように先取りを行う．さらに，TIMES指定の場合には，ループ制御カウンタを更新し，PERFORM 動作を続けるかを判定する．そして，続ける場合には，ループ動作を行うように先取りを行う．
- SEARCH 命令では検索キーが条件を満たすまで表の要素が比較されるものとして，ループを描くように先取りを行う．

2）アドレス展開処理

本プロセッサの主要作業はアドレス展開処理で，オペランドシラブルやデータディスクリプタをもとにデータアドレスの生成を行う．表データについては，アドレス計算に先だって，添字データの上限検査を行う．アドレス計算の後，各オペランドごとに64ビット長の内部ディスクリプタ（図3に一例を示す）を生成し，FIFO キュを介してオペランド取り出しプロセッサへ送る．

3）貯蔵型オペランドの登録

オペランド用データ，指標や添字データについて，命令実行プロセッサがその処理結果を主記憶に貯蔵する前に，後続命令の処理において，命令取り出しプロセッサやオペランド取り出しプロセッサが取り込むのを禁止するために，貯蔵型オペランドを生成するとオペランド情報を先取り制御部へ登録する．一方，命令取り出しプロセッサやオペランド取り出しプロセッサは指標，添字データ，オペランドデータを主記憶から取り込む時には，先取り制御部を用いて処理結果が主記憶に蓄積されたことを確認した後取り込む．もし，処理結果が蓄積されていない時には，そのオペランドの処理が命令実行プロセッサで完了するのを待つ．

4）PERFORM 用ハードウェアスタックとその制御

PERFORM 命令のネストによって生じる複雑な実行シーケンスをサポートするために，専用のハードウェアスタックを備えている[7]．PERFORM 命令を実行すると，スタックには戻りアドレスや PERFORM 対象の最後の命令アドレスなどがプッシュされる．また，PERFORM 対象の命令列を実行している時には，最後の命令を先取りしたか否かをしらべ，その時には，スタックをポップアップする．この動作は普通のスタックで可能な動作である．しかしながら，命令取り出しプロセッサが PERFORM 対象の命令列の最後の命令を処理し，スタックからデータを消去すると問題が生じることがある．たとえば，その後，命令実行プロセッサが最終命令を実行する前に，IF 命令を実行して分岐し，PERFORM 対象の最後の命令を実行しなかった場合には，命令取り出しプロセッサの動作を破棄しなければならない．したがって，この時には，スタックから消去したデータが必要になる．この

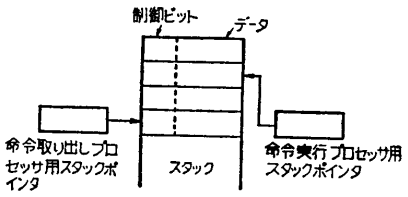


図 4 PERFORM 用ハードウェアスタック
Fig. 4 Hardware stack for PERFORM operation.

ような場合でも正しい順序制御を行えるように，**図4**で示すような2つのプロセッサから見たスタックの状態を示すポインタを独立に備えた専用スタックを設けた．すなわち，命令取り出しプロセッサ用スタックポインタは擬似的なもので，命令実行プロセッサ用スタックポインタが実ポインタとなる．したがって，前記例では，命令実行プロセッサにおいて，IF 命令によって分岐が生じた時には，命令取り出しプロセッサ用スタックポインタを修正することによって，スタック中のデータの保存と命令取り出しプロセッサの制御の破棄を達成できる．

5）ホストプロセッサインタフェースの制御

ホストプロセッサで実行される入出力，通信制御などの命令（ホストプロセッサ呼び出し命令）を検出すると，これまで先取りしている命令列の処理を他のプロセッサが完了したことを確認した後，ホストプロセッサに実行を依頼する．このために，他のプロセッサが実行中であるか否かを示す制御信号が本プロセッサに伝えられている．

5. オペランド取り出しプロセッサ

COBOL マシンは汎用計算機と比べて多くの種類のデータを処理する必要があり，また，異なるデータタイプの間の実行も行う．このために，本プロセッサはデータを取り込み，取り込んだデータの形式を変換することによって命令実行プロセッサが取り扱わねばならないデータの種類を減らし，実行しやすい形に準備する．

1）データの取り込みとデータタイプ変換

数値データは常に取り込まれ，倍精度二進数，符号なしパック十進数などへ変換される．たとえば，加算命令では，**表1**に示される組み合わせ規則により，また，移送命令では，**表2**に示されるように受け手オペランドのデータタイプへ変換される．符号は分離されて，内部ディスクリプタの符号部にセットされる．また，文字列データについては，原則として，32バイト

表1 算術演算のデータ変換規則

Table 1 Data type conversion rule for arithmetic operation.

第1オペランドのタイプ	第2オペランドのタイプ	変換されるタイプ
二進数	二進数	倍精度二進数
二進数	十進数	符号なしパック十進数
十進数	二進数	符号なしパック十進数
十進数	十進数	符号なしパック十進数

表2 移送命令のデータ変換規則

Table 2 Data type conversion rule for MOVE instruction.

受け手オペランドのタイプ	変換されるタイプ
二進数	倍精度二進数
パック十進数	符号なしパック十進数
ゾーン十進数	符号なしゾーン十進数

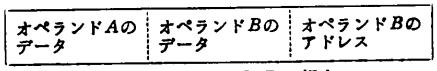
以下のものについてだけ取り込み，何ら変換せずにそのまま移送する．さらに，IF 命令などで，命令実行プロセッサがデータを必要としないような場合（たとえば，POSITIVE, NUMERIC 判定など）にはデータの内容を判定した結果だけを移送する．

2）データの有効検査と内容検査

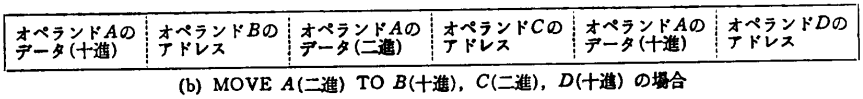
十進データについてはディジット部や符号部が正しいかを検査する．また，編集制御データについては許される編集マイクロオペレータ列であるかをしらべる．また，このような検査以外に，命令実行プロセッサの処理効率を上げることを目的に，取り込んだ内容の検査も行う．たとえば，正，負，ゼロの判定，数字データ，英字データの判定などである．そして，これらの判定結果は内部ディスクリプタに付加されて送られる．

3）オペランド生成とデータの有効活用

命令実行プロセッサの処理を簡単化するために，オペランド用内部ディスクリプタとデータの生成を行う．**図5**(a) は ADD 命令の例を示し，オペランドBについては，データとアドレスの2組の内部ディスク



(a) ADD A TO B の場合



(b) MOVE A(二進) TO B(十進), C(二進), D(十進) の場合

図5 内部ディスクリプタの生成形式

Fig. 5 Internal descriptors generation rule.

リプタを生成する．また，図5(b)の移送命令では，移送処理を1組の単位として，内部ディスクリプタを生成する．したがって，この例では，オペランド B, C, D に対して，オペランド A に関するデータを示す内部ディスクリプタを生成する．そして，オペランド A のデータタイプはそれぞれ B, C, D のタイプに合わせられる．この時，オペランド B のために作られた十進形式のオペランド A が内部に保存される．そして，オペランド D のための十進形式のオペランド A が必要になると，内部に保存されたデータが使用され，オペランド A の主記憶からの取り出しやデータタイプ変換を省略することができる．

6. 命令実行プロセッサ

本プロセッサは前記2つのプロセッサが準備した内部ディスクリプタに基づいて，オペランド取り出しプロセッサが前処理を行ったデータを用いて命令コードで指定された処理を行い，結果を主記憶に保存する．文字列データについては，オペランド取り出しプロセッサが取り込まない33バイト以上のデータについては取り込み処理を行う．

1）十進演算処理と演算レジスタファイル

十進演算は8桁の十進加減算器と，小数点以上と以下を各々32桁保存できる高速演算レジスタファイルを用いて行われる．すなわち，オペランド取り出しプロセッサからデータを受け取る時に，小数点位置を考慮して演算レジスタファイルへ蓄積し，次に，小数点を無視した形で語（8桁）単位で実行する．この時，データが0の場合および値が0である上位や下位の語については演算をスキップする．**図6**は演算レジスタファイルの構造を示す．

2）移送処理ユニット

移送処理は移送命令以外に，演算命令やデータ操作命令などでも頻繁に用いられ，これを高性能に実現する効果は COBOL マシンでは非常に大きい．このために，本プロセッサでは，移送処理専用のハードウェアとして移送処理ユニット（**図7**）を備えている．この移送処理ユニットは，送り手と受け手オペランドを示す内部ディスクリプタに基づいて，桁合せ処理のためのシフト，ゼロやブランクづめ，符号挿入などの処理を統一的に行う．そして，このユニットから主記憶へ書き込むためのデータが語単位で出力される．

3）編集処理ユニット

13種類の編集用マイクロオペレータを実行するもので，編集データ挿入テーブル（スペース，¥，・，，などを蓄積する）や編集制御フラグを用いることによって，固定挿入や浮動挿入などの編集処理を高速に行う．

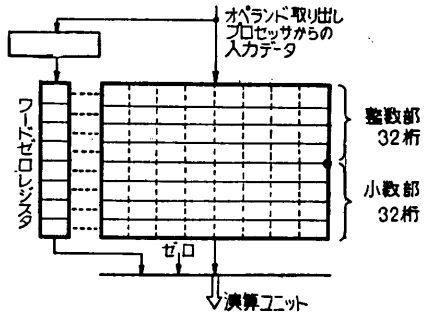


図6 演算レジスタファイルの構造
Fig. 6 Arithmetic register file configuration.

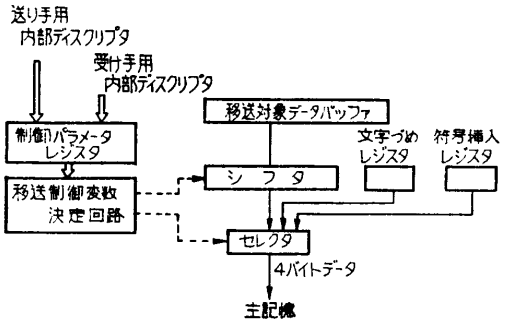


図7 移送処理ユニットの論理構成
Fig. 7 Transfer unit logical structure.

4）先取り制御部の制御

処理結果を主記憶へ蓄積することを完了すると，そのオペランドに対応して登録されているオペランド情報を先取り制御部から消去する．これによって，前記2つのプロセッサがこのオペランドの処理されることを待っている場合には先へ進むことになる．

5）データ操作命令の実行

INSPECT, STRING, UNSTRING などのデータ操作命令はバイト処理用ハードウェアとファームウェアにより実現される．この時，これらの命令を実行するのに必要な多数のオペランド情報は本プロセッサの高速レジスタファイルに蓄積される．また，1つのステートメントがこれらの複数命令に展開された場合に，命令間でチェーニングされるオペランドも高速レジスタファイルに蓄積される．その結果，これらのオペランドデータを主記憶から取り出す必要がなくなり，高速化される．

7. 先取り制御部とメモリインタフェース制御部

命令実行プロセッサが処理結果を主記憶に蓄積する前に，命令取り出しプロセッサが指標や添字データとして，また，オペランド取り出しプロセッサがオペランドデータとして，そのデータを先取りすることを禁止しなければならない．この働きを行うのが先取り制御部で，貯蔵型オペランドに関するアドレスとデータ長をオペランド情報として蓄積する連想メモリで構成される．命令取り出しプロセッサが登録し，命令実行プロセッサが消去し，命令取り出しプロセッサとオペランド取り出しプロセッサが照合を行う．

メモリインタフェース制御部は3つのプロセッサから生じるメモリ要求の制御と仮想記憶制御を行う．3つのプロセッサからの要求を受けつけ，命令実行，オペランド取り出し，命令取り出しプロセッサの順序の優先順位で主記憶にサービスを依頼する．また，仮想記憶制御を高速に行うために，論理アドレスと物理アドレスの対応表を備えている．そして，メモリインタフェース制御部は最近参照されたセグメントについて蓄積するために，主記憶中に存在するセグメントのどれを上記対応表に置くかの管理を行う．

8. ホストプロセッサインタフェース制御

COBOL マシンは ACOS-300 をホストプロセッサとして動作するのに必要なハードウェア，フェームウェア，ソフトウェアインタフェースを備えている．ハードウェアインタフェースにはホストプロセッサの内部バスに直接接続されるインタフェースと主記憶を共用するインタフェースとがある．ファームウェアインタフェースには，ホストプロセッサに内蔵されたCOBOL マシンとのインタフェース制御を行うファームウェア (CSF) と COBOL マシン内のホストプロセッサインタフェース用ファームウェアがある．また，ソフトウェアインタフェースはホストプロセッサ上で動作する COBOL マシンとのインタフェース制御のための COBOL マシンサポートプログラム (CSP) である．**図8**はこれらの関連を示す．

1）実行動作の流れ

1つの COBOL プログラムの実行が始まるとまず，それに対応する CSP が実行される．CSP の中では，前処理を行った後，COBOL マシン起動命令を実行する．本命令は CSF によって解釈実行される．CSF は

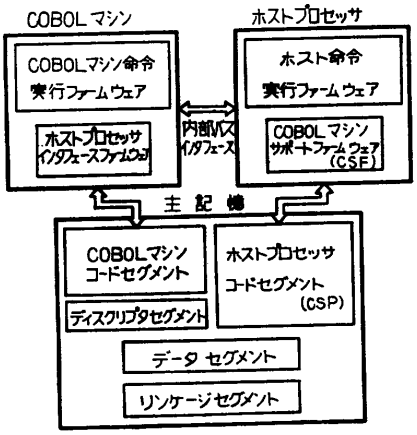


図 8 ホストプロセッサインタフェース
Fig. 8 Host processor interface.

初期処理として，CSP がセットしたレジスタ情報（たとえば，COBOL マシンの命令開始アドレス）を内部バスインタフェースを介して COBOL マシンにセットする．その後，CSF が COBOL マシンの起動を行うと，以後，COBOL マシンはホストプロセッサの動作と並行して実行を続ける．この時，CSF はホストプロセッサまたは COBOL マシンからの割り込みを監視するループに入る．

2）入出力命令などの実行形式

COBOL マシンが入出力命令などを検出すると，ホストプロセッサに処理を依頼する．CSF はこれを受け付けて，必要なパラメータをホストプロセッサに取り込み，COBOL マシンの実行を停止する．そして，COBOL マシン起動命令を一旦終了して，次命令の実行に移る．次命令では，ホスト呼び出し命令の種類を判定し，たとえば入出力命令であれば，CSP の入出力ルーチンを実行する．CSP が実行を完了すると再び COBOL マシン起動命令を実行し，CSF に制御が渡り，COBOL マシンの実行が再開される．

3）プロセス切り変え

ホストプロセッサにおいて割り込みが起り，他のプログラムへのディスパッチが行われる時には，CSF は COBOL マシンの現状態を示す情報を退避した後，他のプログラムへ制御を移す．また，このプログラムに制御が戻った時には，退避した状態を回復することによって処理を再開する．

図9は COBOL マシンの外観図を示す．各プロセッサおよび制御部はビットスライスマイクロプロセッサ (Am 2903) をベースに作られており，各プロセッサのマイクロ命令実行サイクルは 200 ナノ秒である．

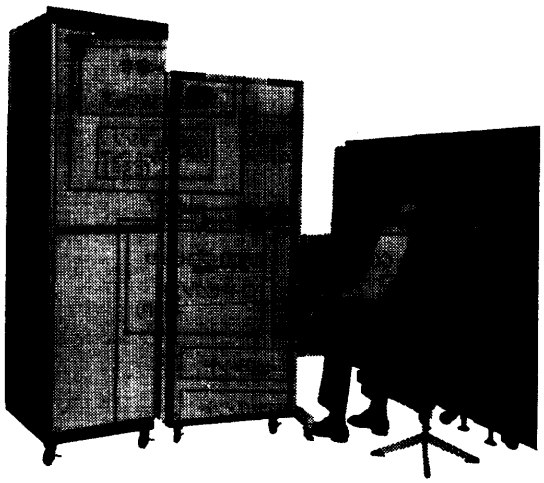


図 9 COBOL マシンの外観
Fig. 9 COBOL machine overview.

3つのプロセッサ（命令取り出し，オペランド取り出しおよび命令実行）はそれぞれ，約1,700，1,500，3,100個の IC で実現されている．

9. むすび

本 COBOL マシンは製作を完了し，汎用計算機 ACOS システム300をベースに COBOL マシンとの結合機構などを組み込んで実現した計算機に接続されて動作している．そして，各種の応用プログラムを COBOL マシンで実行することにより，COBOL マシンと同程度のハードウェア技術で実現された汎用計算機よりも高速に実行されることが明らかになった[8]．特に高性能が実現できた理由として，

- パイプライン制御方式による複合プロセッサ構成をとったこと
- COBOL 処理に専用化したファームウェアおよびハードウェアを内蔵したこと
- マイクロプログラムにより，高機能な命令を高速に実行したこと

などをあげることができる．

なお，詳細な性能評価や設計パラメータの分析結果についての報告は別の機会にゆずることにする．

謝辞 本 COBOL マシンは通産省工業技術院大型プロジェクト「パターン情報処理システム」の一環として開発したもので，本プロジェクトの関係各位に深謝します．また，本 COBOL マシンの開発に当り，日頃有益なご指導を下さった当社ソフトウェア生産技術研究所祢津所長代理および C & C システム研究所箱崎課長に深謝します．

参考文献

1) Wilner, W. D.: Design of the Burroughs B 1700, Proc. FJCC, Vol. 41, pp. 489-497 (1972).
2) Shapiro, M. D.: The criterion COBOL system, Proc. NCC, Vol. 47, pp. 1049-1054 (1978).
3) American National Standard Programming Language COBOL X3.23 1974, American National Standards Institute Inc. (1974).
4) Yamamoto, M., Nakazaki, R., Yokota, M., Hakozaki, K., Umemura, M. and Kumano, K.: Design of a COBOL oriented high level language machine, 3rd USA-Japan Computer Conference, pp. 417-421 (1978).
5) 中崎，山本，梅村，箱崎：COBOL マシンのハードウェア構成，情報処理学会第17回全国大会，pp. 309-310 (1976).
6) 山本，横田，中崎：高級言語ステートメントの実行形態を反映した高級言語マシン，情報処理学会第19回全国大会，pp. 21-22 (1978).
7) 横田，山本，中崎：COBOL マシンにおける命令先取り方式とスタック装置，電子通信学会総合全国大会，pp. 6-43 (1978).
8) 中崎，横田，梅村，山本：COBOL 用高級言語マシンの性能評価，情報処理学会第21回全国大会，pp. 137-138 (1980).

(昭和56年5月25日受付)
(昭和56年7月13日採録)