Arduinoドライブによる Z80 駆動

小出　洋[1,a)]

概要：[概要] 情報処理の黎明期から長足の発展を遂げた各種の処理装置のほとんどはブラックボックス化し，今ではそれらの動作原理を理解するのは著しく困難になりつつある．つまりコンピュータサイエンスの抽象化のギャップは他の分野と比較して大きいため，初学者の学習に困難を生じている．本研究では，初学者が抽象化のギャップを克服する方法について考察し，それを克服するためのひとつの方法として，過去の理解し易いコンピュータの実物を動かせるようにして，初学者がいろいろ試して学習できる手法を 2 種類実装して評価した．そのうちのひとつは実際のコンピュータでメモリに DMA （ダイレクト・メモリ・アクセス）方式であり，ユーザが手動で直接プログラムをメモリに書き込み，クロックを手動で入力して実行する方法と，もうひとつはメモリと制御回路を Arduino で代替する方法である．

キーワード：Z80, マイクロプロセッサ, Arduino, DMA(Direct Memory Access), Arduino 駆動, 学習, 抽象化のギャップ

1. はじめに

情報処理の黎明期から長足の発展を遂げた各種の情報処理のための装置のほとんどは，何段階もの高い抽象化が施されブラックボックス化されている．そのため，個人が現代の情報システムの動作原理を完全に理解することは著しく困難になりつつある．つまり．コンピュータサイエンスにおける抽象化のギャップは他の分野と比較して大きい．そのため，初学者の学習に困難を生じていると考えられる．そこで本研究では，コンピュータサイエンス分野を題材として，初学者が抽象化のギャップを克服するための方法についての考察を行う．この抽象化のギャップを克服するためのひとつの方法として，過去の理解し易いコンピュータの実物を動かせるようにして，初学者がいろいろ試しながら，動作原理を学習できる手法を 2 種類実装して評価した．

そのうちのひとつは実際のコンピュータでメモリに DMA（ダイレクト・メモリ・アクセス）方式であり，ユーザが手動で直接プログラムをメモリに書き込み，クロックを手動で入力して実行する方法（2.1）と，もうひとつはメモリとクロック信号を含む制御回路を Arduino で代替する方法（2.2）である．

1.1 コンピュータサイエンス学習における抽象化のギャップ

著者らは数学では「抽象化のギャップ」は可能な限り排除されていると考えている．小学校から高等学校までの初等中等教育において，体系的に積み上げる形式で体系化されたカリキュラム [1] が存在しており，学習者はステップバイステップで積み上げていけばすべてを理解できるようになっている．学習者は小学校から高等学校までの完全

[1] 九州大学情報基盤研究開発センター
[a)] koide@cc.kyushu-u.ac.jp

な理解が当然とされる教育を受けてきている．

一方，コンピュータサイエンスでは排除すべき「抽象化のギャップ」が多く存在している．コンピュータサイエンスでも基本的な分野・技術は体系化された上で教えられている．ただし，それらの基本的な分野・技術の延長線上にあり，コンピュータサイエンスが実際に産業として応用されている段階にある比較的高度な技術になると，複雑で膨大な内容になってしまう，技術の更新頻度も高く講義内容として定めることは無理である等の理由により，大学などの高等教育におけるカリキュラムでも十分にカバーできている内容になっていない．

コンピュータサイエンスではすべての技術を100%理解することは時間的にも学習量的にも無理であるが，基本的な技術は教えられており，学習者にはその基本的な技術の延長戦上に現在の高度な情報システムが存在していることが理解できる能力が求められている．

2. 抽象化のギャップを克服するための手法

筆者らは 1.1 で述べた，コンピュータサイエンス分野に存在する抽象化のギャップを克服するには，初学者が全貌が理解可能な過去の理解し易いコンピュータの実物を動かしていろいろ試せるようにし，そこから初学者のコンピュータサイエンスの学習に関連する知見を得て抽象化のギャップを克服するための手法を考察することが重要であると考えた．

そこで過去の理解し易いコンピュータとして Z80 を選び，「Z80 コンピュータの DMA による実装（2.1)」と「Z80 コンピュータの Arduino による実装 (2.2)」の二種類を実装したので以下で紹介する．

これらの前提として，多くの情報系大学の学部における学生実験で過去には 8 ビット CPU を利用して機械語や IO 装置の駆動実験を行っていた．いまでは長時間の実験になるという理由のため廃止されたり，縮小されていることが多い．また過去における 8 ビットや 16 ビットの PC の利用経験があげられる．

2.1 Z80 コンピュータの DMA による実装

DMA（Direct Memory Access）は CPU を停止していったんバスから切り離し，他の機器が CPU を介さずにメモリにアクセスできるようにする機能である．現代では高速な通信を実現する為に利用されることが多いが，ここではユーザが手動でメモリにプログラムやデータを直接書き込むことを想定している．ユーザが手で直接メモリにアクセスしたり，CPU の動作を停止したり，開始したりする操作を行うことにより，ノイマン型コンピュータの基本構成である CPU，メモリ，バスの役割を実感として理解することができる（図 1)．

オープンソースになっている Z80 マイクロコンピュータの回路図 [2] を参考にした．このオープンソースの回路図では，アドレスバスの下位 8 ビットのみが配線されており，クロック信号は押しボタンスイッチにより手動で 1 クロックずつ入力する．これを参考にして，16 ビットフルアドレスレコード，実行時にバスの状態を LED で表示，クロック手動入力時のチャタリング対策を強化，RAM を 64KB フル実装する変更を行った上で，実際に作成した（図 2)．

結果，ワンボードとはいえ，すべてを手配線でハンダ付けで作成するのは大変であった，この問題は，プリント基板を利用することにより，解決可能である．また，LED，不揮発性 RAM，C-MOS チップ，トグルスイッチ等の現代の部品は安価かつ高性能であり不具合も無かった．

またクロック信号の入力のチャタリング対策を行うことは必須であり，1 クロックずつ確実に CPU の動作を進めるには工夫が必要であった．しかし，1 クロックずつ CPU の動作を進めることにより，インストラクションフェッチ，命令デコード，オペランドフェッチ等の CPU の動作が実感できることが分かった．

数多くならんでいるスイッチを自在に操作することにより，計算機のすべてを理解して自由に操作できるエモーショナルな操作感を得ることができることが分かった．しかし，このインターフェースで大きいプログラムを試すことは難しい．実際に教育に用いる場合，イニシャルプログラムロー

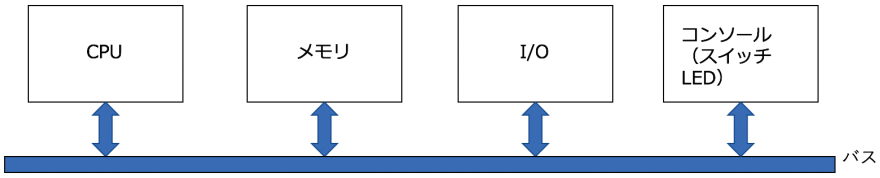

図 1　ノイマン型コンピュータの基本構成.

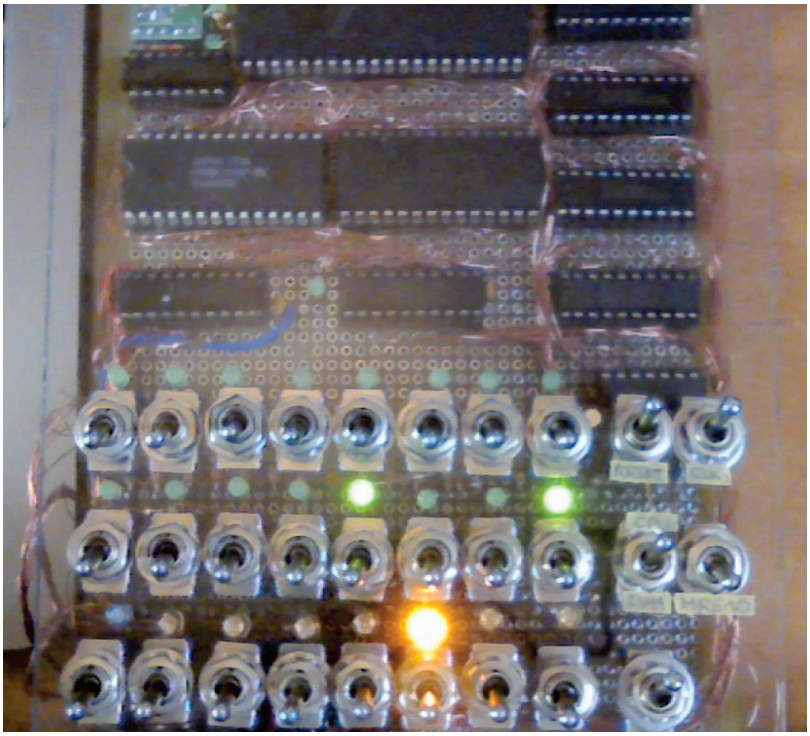

図 2　実際に作成した Z80 マイクロコンピュータ.

ダを入力する演習とすると良いと思われる.

2.2　Z80 コンピュータの Arduino による実装

さらに私たちは現代的な電子パーツを用いたより理解し易い方法を検討した．Arduino は AVR マイコンを搭載したオープンソースハードウェアであり，多くの GPIO 端子を持ち，容易にプログラミング可能な統合開発環境を備えている．Arduino は，GPIO 端子が通常のものより多い Arduino-Mega と呼ばれるタイプを用い，Z80 CPU と直結した．Arduino-Mega にクロック信号の生成およびバス制御信号に対応してメモリの読み書きを模擬するプログラムを記述した．また Arduino のプログラム（「スケッチ」と呼ばれる）内の RAM メモリを表す配列に Z80 の機械語を記述できるようにした（図 3）.

結果，Ardino と Z80 CPU 間の 40 本以下の配線だけで済み，**??**章の DMA による実装よりも手軽に組み立てることができた．ただし，ブレッドボードによる実装はハンダ付けによる配線と比較すると，線をときおり差し込み直す必要がある等，若干不安定なものとなってしまうことが分かった．実際に作成した Arduino 駆動の Z80 の様子を図 4 に示す.

スケッチにおけるメモリ内容を表す配列に機械語を記述可能であるため，DMA による実装と比

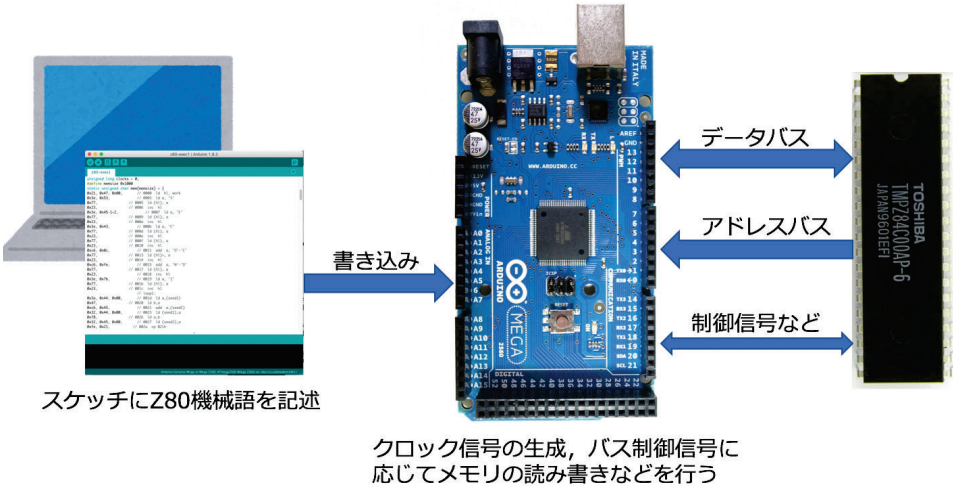

図 3　Arduino により Z80 をドライブ.

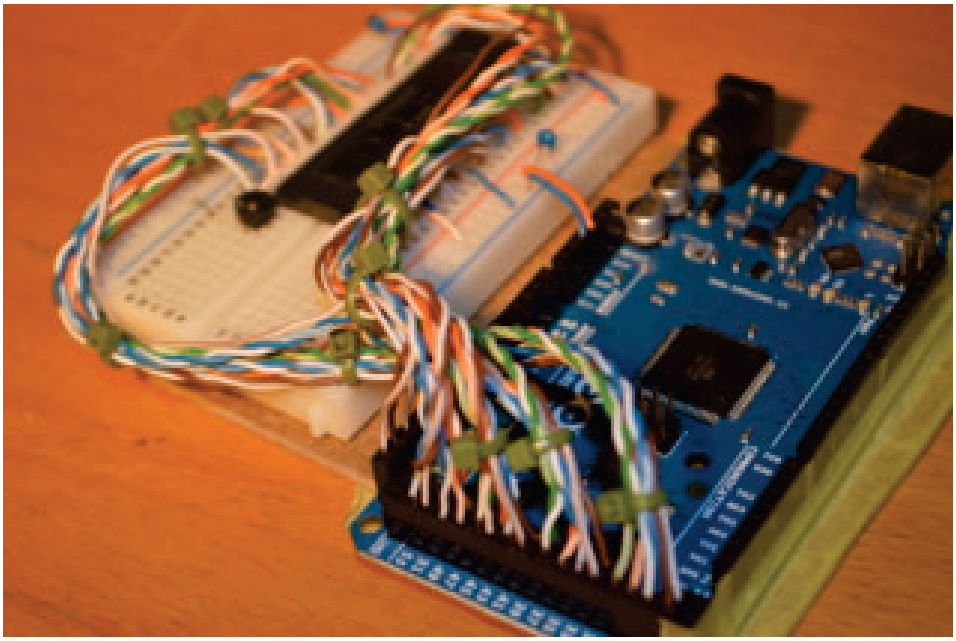

図 4　実際に作成した Arduino 駆動の Z80.

較すると簡単により大きい機械語のプログラムを記述することができる．クロスアセンブラと組み合わせるとより容易に比較的大きい機械語のプログラムをかくことができる．また，実行中の各クロックごとの各バスの信号の様子やメモリの内容の一部を動作させながら Arduino IDE のコンソールに出力すして確認することが可能である（図 5）.

本手法はアンダークロックの下限の存在しない，ユーザが手動クロックで CPU の動作を確認しながら，実行することができる Z80 を使用した．アンダークロックの下限が存在する CPU（MC6809, MOS6502, MC68000 等）を本手法で駆動にはもう少し工夫が必要である.

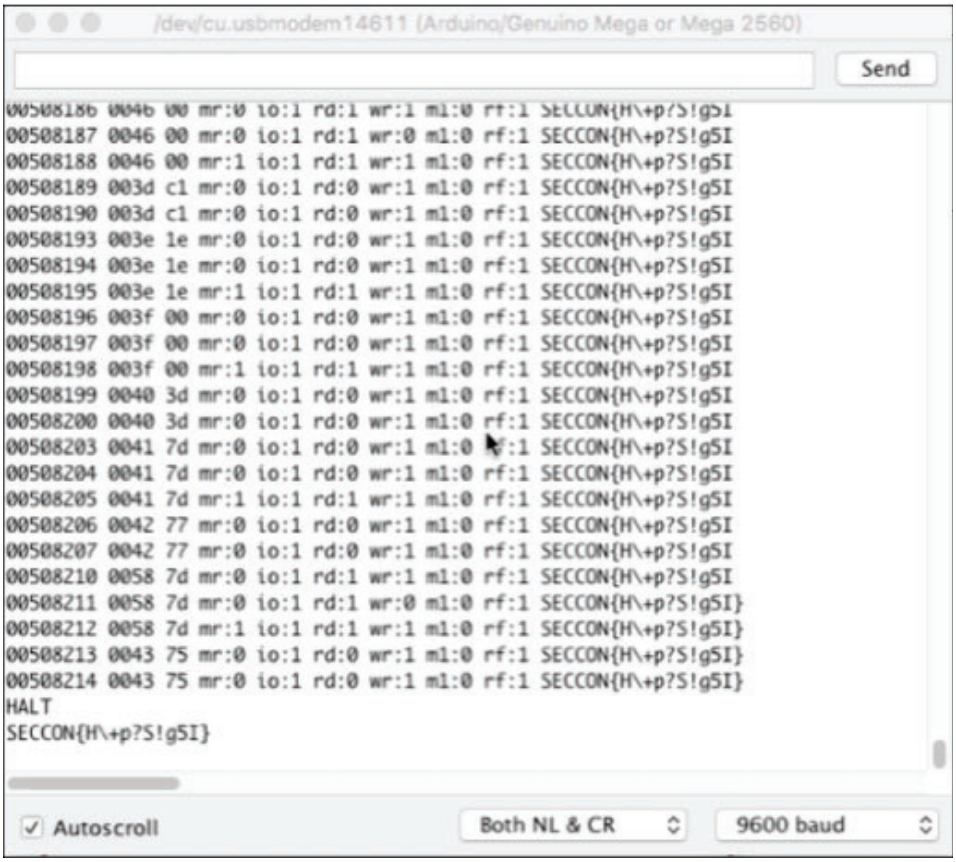

図 5 Arduino 駆動の Z80 の実行中のコンソール.

3. まとめ

本研究では，初学者が学習する際，困難を生じるコンピュータサイエンスにおける抽象化のギャップを克服するための方法についての考察を行い，抽象化のギャップを克服するためのひとつの方法として，過去の理解し易いコンピュータの実物を動かせるようにして，初学者がいろいろ試して学習できる手法を 2 種類実装して評価を行った．結果，Arduino 駆動による方法は，手軽に CPU の動作を確認することができ，今後の検討が可能な方法であることが分かった．さまざまな過去の理解し易い CPU に本提案手法を適用するにはもう少し工夫が必要になるが，ある程度の大きさの機械語プログラムを試すことができることもわかった．コンピュータサイエンスにおける，抽象化のギャップを克服するための，一助になる可能性があることが分かった．

質疑応答

講演後に以下の質疑応答があった．質問者の敬称を略させていただいた．

岸本：(68000 で最低クロック周波数の制限があり同様の仕組みは難しいという文脈において,) 68000 ではメモリ待ちの仕組みがあり，同様の手法が可能ではないか．

小出：ワークステーションの仮想記憶を実現するために良く利用されていた仕組みであり，試してみようとしている．

田中（インターフェィス）: 実現の手応えはどうか．

小出：手応えはある．

新谷（福山大）: 学生実験の対象者はどのような人

たちか.
小出：計算機システムを理解したいファイ学生を想定している.
内藤（富士通特機）: 機械語や電気信号のレベルを変換すれば，昔のリレー式計算機も利用できるのではないだろうか.
小出：利用できると考えられる.
丹治（デンパン）: FPGA を使ってより教育的な計算機にした方が良いではないだろうか.
小出：実際の CPU を使いたいのと，FPGA にしてしまうと何もかも詰め込めてしまい，そこで抽象度が上がってしまうことがあり，目に見えるという理解のし易さとのトレードオフが存在する.
岸本：x86 もそうだがリトルエンディアンの CPU はそこでつまずいてしまわないか．小出：特に気になることはなかったが，もっと検討が必要である.
川端（広島市立大）: この課題を学生実験にどう応用したいのか.
小出：学生実験に落とし込むにはもっと検討が必要であると考えている.

参考文献

[1] 文部科学省: 新学習指導要領における算数・数学教育の内容, http://www.mext.go.jp/b_menu/shingi/chuk-yo/chukyo3/013/siryo/04071301/003.htm, 2018 年 11 月 16 日確認.
[2] zeta256, https://github.com/jmacarthur/zeta256, 2018 年 11 月 26 日確認.