Nintendo Entertainment System ドキュメント

バージョン1.0
2004年8月 パ
トリック・デ
ィスキン

巻頭言

アブストラクト

Nintendo Entertainment System（NES）は、1980年代に世界で最も広く普及したビデオゲーム機です。1983年に発売されてから1995年に製造中止になるまで、このゲーム機はかつてないほど多くの家庭にゲームをもたらし、今日のビデオゲーム業界の道を切り開きました。

ファミコン以降、技術は劇的に進歩しましたが、多くの優れたゲームはファミコンでしか発売されなかったため、最新のシステムではプレイできませんでした。しかし、エミュレーションとは、あるシステムの動作をシミュレートして、そのシステム用に作られたソフトウェアを最新のシステムで使用できるようにするもので、そのおかげでこれらのゲームは生き残り、プレイされ続けています。

このドキュメントでは、ファミコンのハードウェアと、ファミコンで使用されるいくつかのデバイスについて説明しています。また、エミュレーションとその問題についても簡単に触れています。このドキュメントの内容の多くは、以前に[1]に掲載されています。

このドキュメントでは、16進法と2進法のナンバリングシステムを使用しています。読者はこれらの番号体系についてある程度の知識を持っていると思われますが、いくつかの問題については付録Aで簡単に説明しています。

謝辞

このドキュメントに記載されている情報は、NESエミュレーションに関わる多くの人々の作業に基づいています。参考文献」に記載されているすべての文書の著者に感謝しますが、特に以下の方々に感謝します。

- Andrew John Jacobs氏には，6502プロセッサに関する貴重な情報を提供していただきました[2]，[3]，[4]。
- Chris Covell氏による「NES Technical / Emulation / Development FAQ」[5]。
- Comprehensive NES Mapper Document」のFirebug [6]。
- Jeremy Chadwickによる「Nintendo Entertainment System Documentation」[7]。
- The Skinny on NES Scrolling」[8]のルーピー。
- Marat Fayzullin氏による「Nintendo Entertainment System Architecture」[9]。
- nesdev.parodius.comに関わる全ての人。

1 - はじめに

1.1 任天堂エンターテイメントシステムの歴史

1889年、山内房次郎は任天堂コッパイを設立し、京都で日本のトランプである花札の製造を開始した[10]。山内溥が社長に就任した1950年には、任天堂は西洋と日本の両方のトランプを製造するメーカーとして成功していた。1963年、何度かの社名変更を経て、「任天堂株式会社」に落ち着いた。Ltd.(NCL)となった。(NCL）となった。)1970年までに同社は電子ゲームを製造し、1973年にはボーリングに代わる主要な娯楽として期待されたレーザークレイシューティングシステムを発表した[11]。

ノーラン・ブッシュネルはユタ大学の学生だったが、コイン式のコンピューターゲームのアイデアを最初に思いついたのは1972年だった。1972年に発売された「ポン」は瞬く間にヒットし、アーケードゲームが次々と発売されるきっかけとなった。ブッシュネルの会社であるアタリ社は、この成功を再現するために、家庭でゲームをプレイできるシステムを発売しようと考えた。1976年までに、いくつかの会社が家庭用ゲーム機を発売しようとして失敗していた。ブッシュネルは、アタリ社にはゲーム機を製造するための資本がないことを認識しており、会長の地位を維持したまま会社をワーナー・コミュニケーションズに売却した[12]。

1977年、アタリ社は8ビットゲーム機「アタリ・ビデオ・コンピュータ・システム（VCS）」を発売し、1980年に発売された家庭用ゲーム機版「スペースインベーダー」も手伝って、家庭用ゲーム機の市場を開拓することに成功した。ブッシュネルはワーナーの方向性に納得がいかず、1978年に会社を去った。

1979年、任天堂はアーケードゲーム市場への参入を初めて試みたが、1981年にはその成功は限られたものとなっていた。山内博史は、任天堂のグラフィックアーティストである宮本茂に新しいゲームのデザインを依頼した。その結果、生まれたのが「ドンキーコング」だった。プレイヤーは大工の「ジャンプマン」を操り、捕らえられた少女を大きな猿の「ドンキーコング」から救い出すという内容だった。ジャンプマンは、任天堂が新たに設立したアメリカの子会社、Nintendo of America Inc.（後のNOA）の大家の名前にちなんで、マリオと名付けられた。(山内の義理の息子である荒川稔が経営していた。

1982年になると、サードパーティの開発により、アタリ社のVCS向けに標準以下のゲームがいくつか発売され、他のゲーム機との競争で市場が飽和状態になっていた。1984年になると、業界は莫大な損失を被り、ほとんどの製品ラインが製造中止となった。

一方、任天堂は、アーケード市場や家庭用ゲーム機「カラーテレビゲーム6」で成功を収めていた。日本の家庭用ゲーム機市場はまだ好調で、山内は、質の高いゲームと改良されたハードウェアを組み合わせて、他社よりも安い価格（ゲームで利益を得る）で販売すれば、任天堂は市場のリーダーになれると考えていた。

図1-1.Nintendo Entertainment SystemとFamicom[13]です。

日本で大成功を収めたファミコンは、1983年に荒川實氏がアタリにアメリカでの生産を依頼した。しかし、アタリには事業を進めるだけの資金力がないことが明らかになり、この契約は破談となった。アタリはワーナーに売却され、任天堂はファミコンを「NINTENDO ENTERTAINMENT SYSTEM（NES）」という名称でアメリカで生産・販売することになった。ファミコンは、図1-1に示すように、欧米の子供たちにアピールするためにデザインを一新した。

1985年、アメリカでファミコンが発売されたが、前年の不況の影響で小売店の抵抗があった。スーパーマリオブラザーズ」「ゼルダの伝説」「メトロイド」など、任天堂が開発したゲームに加え、サードパーティのソフトにも厳しい品質管理が行われた結果、このゲーム機は大成功を収めました。

1987年、ファミコンはアメリカで最も売れたおもちゃとなり、『ゼルダの伝説』はファミコンのゲームとしては初めて100万本の売り上げを達成した。アメリカだけでも、『スーパーマリオブラザーズ3』の収益は700万本以上、日本では400万本が販売され、5億円を超えた[14]。1991年、任天堂は5,000人の従業員一人当たり約150万円の収入を得ていた。1990年代初頭の同社の利益は、アメリカの映画産業の利益を上回っていた。任天堂のアメリカ文化への影響は大きく、1990年の調査では、マリオはミッキーマウスよりも子供たちに認知されているという結果が出ている。

セガは1989年に16ビットの「ジェネシス」（欧州では「メガドライブ」）を発売し、「ソニック・ザ・ヘッジホッグ」のヒットにより、このゲーム機は人気を博した。同年、任天堂は携帯型ゲーム機「ゲームボーイ」の発売で忙しかったが、1990年には「スーパーファミコン」で16ビット市場に参入した。1991年にアメリカで「Super Nintendo Entertainment System（SNES）」として発売されたこのゲーム機は、ファミコンのハードウェアとの互換性がないため、古いシステムからの脱却を意味していた。

図1-2.1993年に発売されたリデザインされたファミコン[15]。

1993年、任天堂はデザインを一新したファミコンを発売したが（図1-2）、1994年末に最後のファミコンゲーム「ワリオの森」が発売され、1995年に正式に製造中止となった[16]。この時までに、全世界で6000万台以上のファミコン本体と5億本以上のゲームが販売されていた。

SNESは65816プロセッサを搭載しており、NESの6502プロセッサとほぼ互換性があった

。しかし、新システムのグラフィックとサウンドには互換性がなかった[5]。このため、旧システム用に作られたゲームを後継システムで動かすことはできなかった。その結果、ファミコン用に作られたソフトは、そのシステムを持っていない人には使われなくなってしまった。

ファミコンを持っていても、ソフトを使えない人が多いのです。また、すべてのハードウェアには寿命があり、最終的にはゲームを遊べるファミコン本体がなくなってしまいます。また、ゲームの進行状況を保存するために電池式のRAMが搭載されていましたが、任天堂は電池の寿命を5年と予測していました。ここでは、ファミコンゲームを継続して使用するための複数の方法をご紹介します。

1.2 変換

コンピュータシステムの正確な実装は異なるが、原理の多くは同じである。ファミコン用に書かれた命令は、PCでは理解できないので実行できません。しかし、PCには同等の命令が存在する可能性があります。そのため、ソフトを別のシステム用に書き換えて、オリジナルのグラフィックやサウンドを再現することができるのです。このようにソフトウェアを変換することは、本質的にはシミュレーションである[17]。ソフトウェアはオリジナルと同じ動作をしているように見えますが、実装は全く異なる可能性があります。

ソフトウェアをコンバートすることで、対象となるアーキテクチャに合わせて制作されているため、出来上がったソフトウェアの性能が向上するというメリットがあります。しかし、この作業は時間がかかり、ゲームごとに行わなければなりません。

1.3 エミュレーション

エミュレーションとは、ハードウェアをシミュレートして、そのために開発されたソフトウェアを互換性のないシステムで使用できるようにするプロセスである。以下の定義は、British Computer Societyによるもので、[18]から引用しています。

> 「エミュレーションとは、非常に精密なシミュレーションであり、シミュレートしている状況の動作を正確に模倣しなければならない。エミュレータは、ある種類のコンピュータを、あたかも別の種類のコンピュータのように動作させることができる。"

ビデオゲーム業界では、新しいシステムが発売される前に、開発者がそのシステム用のソフトウェアを書き始めるために、エミュレーションがよく使われる。しかし、古いシステムを継続して使用するためにも使用されることがあります。

ハードウェアエミュレーションとは、オリジナルと互換性のあるハードウェアを使ってシステムを作ることです。ファミコンの場合は、互換性のあるプロセッサを使ってシステムを作り、オリジナルのゲームカートリッジを遊べるようにすることができます。互換性が確保されていれば、この手法でも十分な性能を得ることができるが、システムを構築するために必要なスキルやリソースを持っている人は少ない。

ハードウェアシミュレーションソフトウェアを使えば、この技術を半永久的に実現することができます。詳細なハードウェア設計をシミュレーションできるソフトウェアがあり、これを使えば、実際の実装を行わなくても、設計からシステムを再現することができます。このようなシステムは[19]に記載されています。

ソフトウェア・エミュレーションでは，与えられたシステムの機能をエミュレートするソフトウェアを作成する必要がある．ソフトウェア・エミュレーションには3つのアプローチがある[17]。

- 解釈とは、エミュレートされているシステムの次の命令を読み込み、それをターゲットアーキテクチャの命令（または複数の命令）に変換して実行することです。正確ではありますが、実行時に変換するため、ターゲットシステムの速度が低い場合には、元のシ

ステムに比べて性能の低下が顕著になります。

- 静的翻訳とは、ソースプログラム全体を読み込み、ターゲットシステム用に翻訳することで、ターゲットシステムで実行可能なプログラムを作成することです。しかし、プログラムを静的に解析しただけでは、そのプログラムがどのように実行されるかを判断できない場合があります。例えば，分岐命令は，実行時にしか判断できないメモリの場所の内容に依存することがよくあります[20].

- 動的翻訳は、静的翻訳とほぼ同じように、プログラムを実行しながら行われます。これにより，分岐命令やジャンプ命令を考慮し，正確なコードを生成することができます[20].

ファミコンは、おそらく最も広くエミュレートされているゲーム機であり、様々な品質のエミュレーターがすでに利用可能です。NESのエミュレータを作成することは、システムがどのように動作するかを詳細に理解する必要があり、非常に困難なプロジェクトです。利用可能なNESエミュレータの包括的なリストは[21]にありますが、これらの多くは廃止されています。エミュレータを書くための基本的なことは[17]と[22]に書かれていますが、どちらも特にNESに焦点を当てています。

1.4 リーガル

エミュレーションは、法的にはグレーゾーンのようなものと考えられています。エミュレータは、開発に使用されたすべての情報が合法的に入手され、いかなるプロプライエタリコードも含まれていなければ、違法ではありません。しかし、ライセンスを所有していないソフトウェアを実行することは違法です。

ファミコンのゲームをコピーするには、適切なハードウェアが必要です。このコピー機は、ゲームカートリッジの内容をディスクにダンプし、コンピュータからアクセスできるようにするものです。複写機にはさまざまな種類があり、その機能もさまざまです。図1-3は、USBポート経由でカートリッジの内容をPCにコピーするChameleonNESです。通常、著作権法ではバックアップコピーが認められていますが、ファミコンのように永久保存可能な半導体チップに格納されたゲームには適用されません。これらのコピー機器は違法です。

図1-3.ChameleonNESのカートリッジコピー機[23]。

エミュレータユーザーの多くは、必要なコピーハードウェアを入手できないため、インターネットからゲームをダウンロードしています。これらのウェブサイトは通常、オリジナルのゲームを所有している場合にのみファイルをダウンロードできる、または24時間以内に削除するという契約でカバーしています。これにより、プロセスは合法的に見えますが、コピーが違法である以上、明らかに違法です。仮にファミコンゲームのバックアップコ

ピーが法律で認められていたとしても、コピーを作成して使用できるのは元の所有者だけなので、インターネットでファイルをダウンロードすることは違法となります。

これは、もともとファミコン本体で作られたゲームに限った話です。その後、多くの開発者が制作したゲームがインターネット上で自由に公開されています。これらをダウンロードすることは問題ありません。オリジナルのファミコンゲームについては、開発者が使用を許可するか、制作から75年後に著作権が消滅するまでは違法となります。

任天堂はエミュレーションに強く反対しています。任天堂は、同社のNintendo 64コンソール用のエミュレーターであるUltraHLE [24]の開発者に対して、著作権を侵害していると主張して行動を起こした。UltraHLEがリリースされた当時、NINTENDO64はまだ使用されていたので、任天堂の反対は理解できた。ニンテンドー64のエミュレーターの存在は、任天堂の収入を脅かすものだったが、もはや生産されておらず、任天堂がもはや収益を上げていないシステムについては、法律で例外を設けるべきだと考える人も多い。エミュレーターが古いゲームを存続させることができるにもかかわらず、任天堂は古いゲームの著作権を放棄しないため、エミュレーターの使用は依然として違法である。

エミュレーションの法的問題の詳細については、この問題に関する任天堂自身のFAQを参照してください[25]。これは非常に偏った内容で、エミュレーションの合法的な使用法を無視しています。この件に関する詳細な回答と説明については、[26]をご覧ください。

1.5 NESハードウェアの概要

山内宏の「他社よりも安いゲーム機を作れ」という指示により、任天堂は旧式のCPUを採用することになった。16ビットであれば問題ないのだが、価格を抑えるために、1975年にMOSテクノロジー社が開発した8ビットの6502プロセッサを採用した。プログラムを実行するには十分だが、グラフィックスを生成することはできないので、グラフィックスの計算と表示を行う専用のPPU（Picture Processing Unit）として、2つ目のチップを使用することにしたのだ。図1-4は、マザーボードの上面にCPUとPPUを表示したものである。

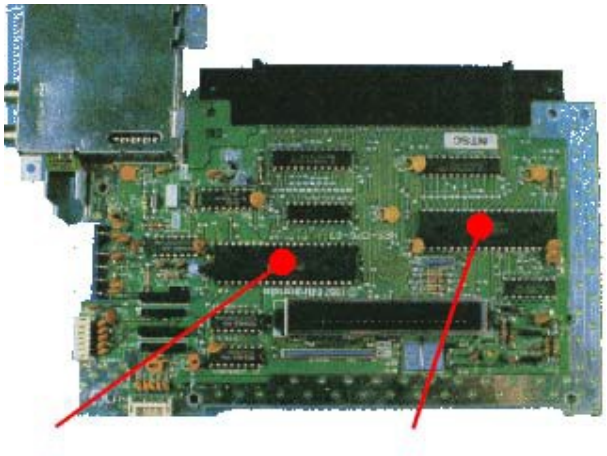


図1-4.NESのマザーボード[27]。

任天堂は、チップに求められる基本的な機能を設計していたが、このような高度にカスタマイズされたチップを低価格で製造してくれる会社を探すのは困難だった。リコーは、任天堂が300万チップの発注を保証してくれたことで、チップの製造を承諾した。1986年末には、任天堂はリコーの最大の顧客となり、同社の半導体売上の60～70%を占めるようになった[10]。CPUの機能については第2部で、PPUの機能については第3部で説明する。

どちらのチップにもRAMという内部メモリが搭載されている。ゲームは通常、ゲームカートリッジ内のROMチップに格納されており、カートリッジがシステムに挿入されると、CPUからアクセスできるようになっていた。ゲーム用のハードウェアについては第4回で紹介する。

ファミコンでは、プロセッサが他のコンポーネントであるPPUや入力デバイスと通信するために、メモリマップドI/Oを使用していました。メモリマップドI/Oとは、データが

は、メモリ内の特定の場所への書き込みを介してデバイスに転送することができます。入力デバイスについてはパート5で説明していますが、メモリマップドI/Oの機能については、このドキュメント全体、特に付録Bで説明しています。

2 - セントラル・プロセッシング・ユニット

2.1 2A03概要

リコーは、6502をベースにしたNMOSプロセッサ「2A03」を製造した。標準の6502との違いは、CPUだけでなくpAPU（pseudo-Audio Processing Unit）として音声を扱うことができることと、1桁を4ビットで表現するBCD（Binary Coded Decimal）モードがないことである。2A03は、図2-1に示した標準的な6502と同じ命令セットを使ってプログラミングを行う。6502はリトルエンディアン方式を採用しており、アドレスは最下位バイトから順にメモリに格納される。例えば、アドレス$1234は、メモリ位置xに$34、メモリ位置(x+1)に$12と格納される。

図2-1.6502プロセッサ[28]。

2.2 CPUメモリーマップ

図2-2は、CPUがバスを使ってメモリにアクセスする様子を示している。メモリは、カートリッジ内のROM、CPUのRAM、I/Oレジスタの3つの部分に分かれている。アドレスバスは、必要な場所のアドレスを設定するために使用されます。コントロールバスは、要求が読み出しなのか書き込みなのかを各コンポーネントに知らせるために使用されます。データバスは、選択されたアドレスへのバイトの読み出しまたは書き込みに使用されます。なお、ROMはリードオンリーで、バンク切り替えのためにMMCを介してアクセスします。I/Oレジスタは、システムの他のコンポーネント、PPU、制御デバイスとの通信に使用されます。

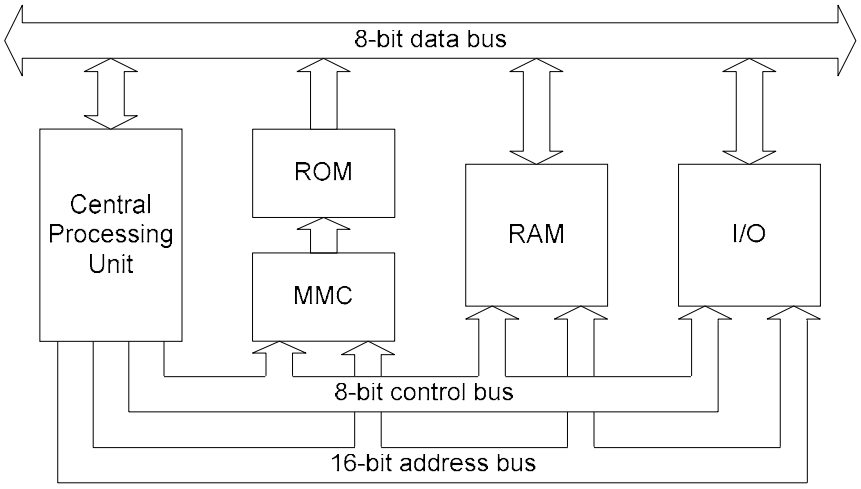


図2-2.プロセッサの図。

2A03は16ビットのアドレスバスを持ち、$0000～$FFFFのアドレスを持つ64KBのメモリをサポートしていたのだ。図2-5は、ファミコンで使用されているCPUメモリマップで、メモリの配置を示しています。左側のマップは主要な部分を簡略化したもので、右側のマップはさらに分割したものです。

ゼロページとは、メモリの最初のページである$0000～$00FFの範囲のアドレスを指し、特定のアドレッシングモードでは、より高速な実行を可能にするために使用されます[4]。メモリの位置
0000-$07FFは$0800-$1FFFに3回ミラーリングされます。つまり、例えば、$0000に書き込まれたデータは、$0800、$1000、$1800にも書き込まれるということです。メモリマップされたI/Oレジスターは、$2000-$401Fに配置されています。2000-$2007の位置は、$2008-$3FFFの領域に8バイトごとにミラーリングされ、残りのレジスタもこのミラーリングに従います。SRAM（WRAM）はSave RAMで、ゲームのセーブデータを保存するカートリッジ内のRAMにアクセスするためのアドレスである。

8000から先はカートリッジPRG-ROMに割り当てられたアドレスです。PRG-ROMの16KBバンクが1つしかないゲームでは、$8000と$C000の両方にPRG-ROMをロードします。これは、ベクターテーブルが正しいアドレスに配置されるようにするためです。16KBのPRG-ROMバンクが2つあるゲームでは、1つを$8000に、もう1つを$C000にロードします。2つ以上のバンクを持つゲームでは、メモリマッパーを使用して、どのバンクをメモリにロードするかを決定します。メモリマッパーは、特定のアドレス（またはアドレスの範囲）に対するメモリの書き込みを監視し、そのアドレスが書き込まれると、バンクスイッチを実行します。詳細はメモリマッパーによって異なり、詳細は付録Dに記載されています。

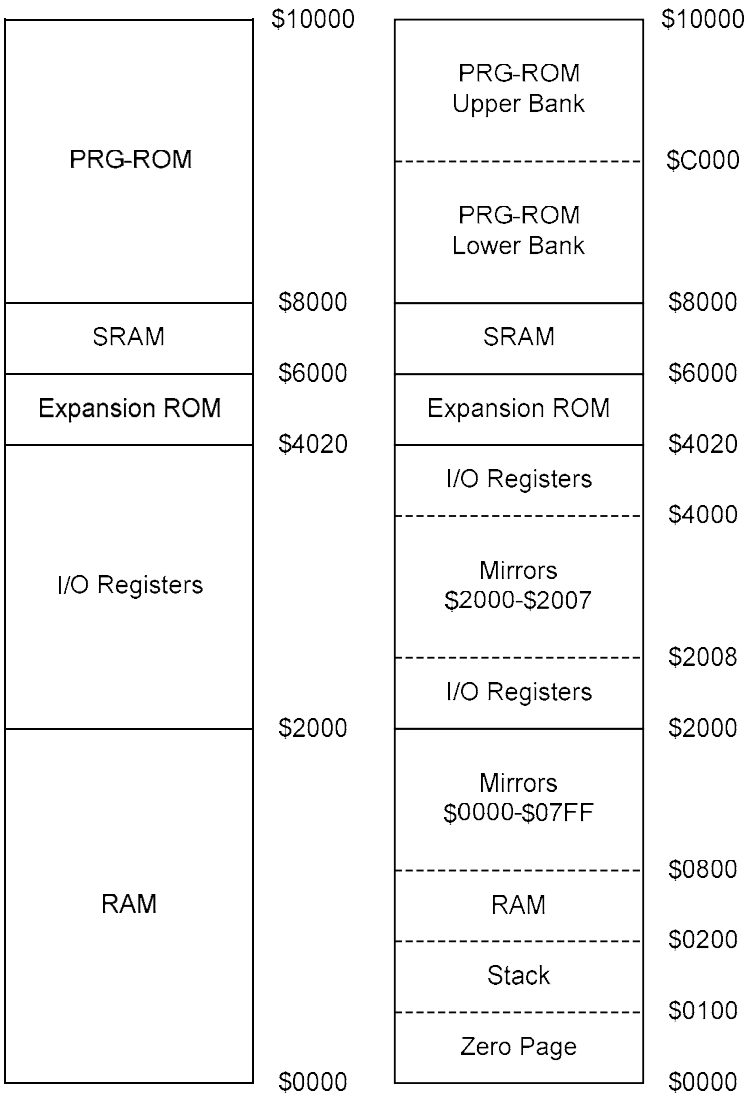


図2-3.CPUのメモリーマップ。

2.3 レジスター

6502は、他のプロセッサに比べてレジスタの数が少ない。プログラムカウンタ、スタックポインタ、ステータスレジスタの3つの特別目的レジスタがあり、それぞれ特定の用途に使用されます。また、データや制御情報を一時的に保存するために、アキュムレータとインデックスレジスタ（XとY）の3つの汎用レジスタを備えています。

2.3.1 プログラムカウンター（PC

プログラムカウンタは、次に実行される命令のアドレスを保持する16ビットのレジスタです。命令が実行されると、プログラム・カウンタの値が更新され、通常はシーケンス内の次の命令に進みます。プログラムカウンタの値は、分岐命令やジャンプ命令、プロシージャコール、割り込みなどの影響を受けることがあります。

2.3.2 スタックポインター（SP

スタックは、メモリ位置$0100～$01FFに配置されています。スタック・ポインタは8ビットのレジスタで、$0100からのオフセットとして機能します。スタックはトップダウンで動作するので、バイトがスタックにプッシュされると、スタック・ポインタはデクリメントされ、バイトがスタックからプルされるとスタック・ポインタはインクリメントされます。スタックオーバーフローは検出されず、スタックポインタは$00から$FFまで回り込みます。

2.3.3 アキュムレータ（A

アキュムレータは、算術演算や論理演算の結果を格納する8ビットのレジスタです。アキュムレータには、メモリから取得した値を設定することもできます。

2.3.4 インデックスレジスタX（X

Xレジスタは、8ビットのレジスタで、通常、カウンタや特定のアドレッシングモードのオフセットとして使用されます。Xレジスタは、メモリから取得した値を設定したり、スタックポインタの値を取得または設定するために使用することができます。

2.2.5 インデックス・レジスタY（Y

Yレジスタは、8ビットのレジスタで、Xレジスタと同様に、カウンタやオフセットを格納するために使用されます。Xレジスタとは異なり、Yレジスタはスタックポインタに影響を与えません。

2.3.6 プロセッサのステータス（P

ステータスレジスタには、命令が実行されるとセットまたはクリアされる1ビットのフラグが多数含まれています。

- キャリーフラグ(C) - 最後の命令で、ビット7からのオーバーフローまたはビット0からのアンダーフローが発生した場合、キャリーフラグが設定されます。例えば、255 + 1を実行すると、キャリービットが設定されている場合、答えは0になります。これにより、1バイト目で計算を行い、キャリーを格納し、2バイト目で計算を行う際にそのキャリーを使用することで、8ビットよりも長い数値の計算を行うことができます。キャリーフラグは、SEC（Set Carry Flag）命令でセットでき、CLC（Clear Carry Flag）命令でクリアできます。
- ゼロフラグ(Z) - 最後の命令の結果がゼロであった場合、ゼロフラグが設定されます。例えば、128 - 127 はゼロフラグをセットしませんが、128 - 128 はセットします。
- 割り込み禁止(I) - 割り込み禁止フラグは、システムがIRQに応答しないようにするために使用できます。SEI(Set Interrupt Disable)命令でセットされ、CLI(Clear Interrupt Disable)命令が実行されるまでIRQは無視されます。
- 10進モード(D) - 10進モードフラグは、6502をBCDモードに切り替えるために使用されます。しかし、2A03はBCDモードをサポートしていないため、フラグを設定してもその値は無視されます。このフラグは、SED(Set Decimal Flag)命令で設定でき、CLD(Clear Decimal Flag)でクリアできます。
- ブレークコマンド(B) - ブレークコマンドフラグは、BRK(Break)命令が実行され、IRQが発生したことを示すために使用されます。
- オーバーフローフラグ（V） - 前の命令で無効な2の補数の結果が得られた場合、オーバーフローフラグが設定されます。これは、正の結果を期待していたのに負の結果が得られた、またはその逆であることを意味します。例えば、2つの正の数を足すと正の答えが得られるはずです。しかし、64 + 64は、符号ビットのために-128という結果になり

ます。そのため、オーバーフローフラグが設定されます。オーバーフローフラグは、ビット6とビット7の間、およびビット7とキャリーフラグの間のキャリーの排他的論理和を取ることで決定されます[29]。2の補数の説明は付録Aにあります。

- ネガティブフラグ（N） - バイトのビット7はそのバイトの符号を表し、0が正、1が負となります。この符号ビットが1の場合、負のフラグ（符号フラグとも呼ばれる）が設定されます。

フラグは，図2-3に示す順序でステータスレジスタに配置されています。ステータスレジスタのビット5は未使用です。

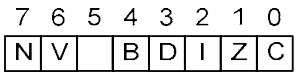


図2-4.ステータスレジスタのレイアウト

2.4 割り込み

割り込みは、コードの標準的な逐次実行を妨げ、プロセッサに割り込みへの対応をさせます。割り込みは通常、注意を必要とするハードウェアによって生成されますが、ソフトウェアによって起動することもできます。NESには、NMI、IRQ、リセットの3種類の割り込みがあります。割り込みが発生したときにジャンプするアドレスは、プログラムコード中のベクターテーブル（$FFFA～$FFFF）に格納されています。割り込みが発生すると、システムは以下の動作を行います[30]。

1. 割り込み要求が発生したことを認識します。
2. 現在の命令の実行を完了する。
3. プログラムカウンタとステータスレジスタをスタックにプッシュします。
4. 割り込み禁止フラグを設定して、さらなる割り込みを防止します。
5. 割り込み処理ルーチンのアドレスをベクターテーブルからプログラムカウンタにロードする。
6. 割り込み処理ルーチンを実行します。
7. RTI（Return From Interrupt）命令を実行した後、プログラムカウンタとステータスレジスタの値をスタックから引き出します。
8. プログラムの実行を再開します。

IRQ（Maskable Interrupt）は、特定のメモリマッパーによって生成されます。IRQは、割り込み禁止フラグが設定されている場合、プロセッサによって無視されます。IRQは、BRK(Break)命令を使用してソフトウェアによりトリガすることができます。IRQが発生すると、システムは$FFFEと$FFFFにあるアドレスにジャンプします。

NMI（Non-Maskable Interrupt）とは、各フレームの終わりにV-Blankが発生したときにPPUが生成する割り込みの種類です。NMIは、ステータスレジスタの割り込み禁止ビットの影響を受けないため、発生すると常に実行が中断されます[31]。しかし、PPU制御レジスタ1（$2000）のビット7がクリアされていれば、NMIのトリガを防ぐことができます。NMIが発生すると、システムは$FFFAと$FFFBにあるアドレスにジャンプします。NMIの処理を図2-4に示します。

リセット割り込みは、システムの初回起動時とユーザーがリセットボタンを押した時に発生します。リセットが発生すると、システムは$FFFCにあるアドレスにジャンプして$FFFDです。

システムはリセット要求に最も高い優先順位を与え、次にNMI、最後にIRQと続きます[7]。NESの割り込みレイテンシは7サイクルで、割り込みハンドラの実行開始までに7CPUサイクルかかります。

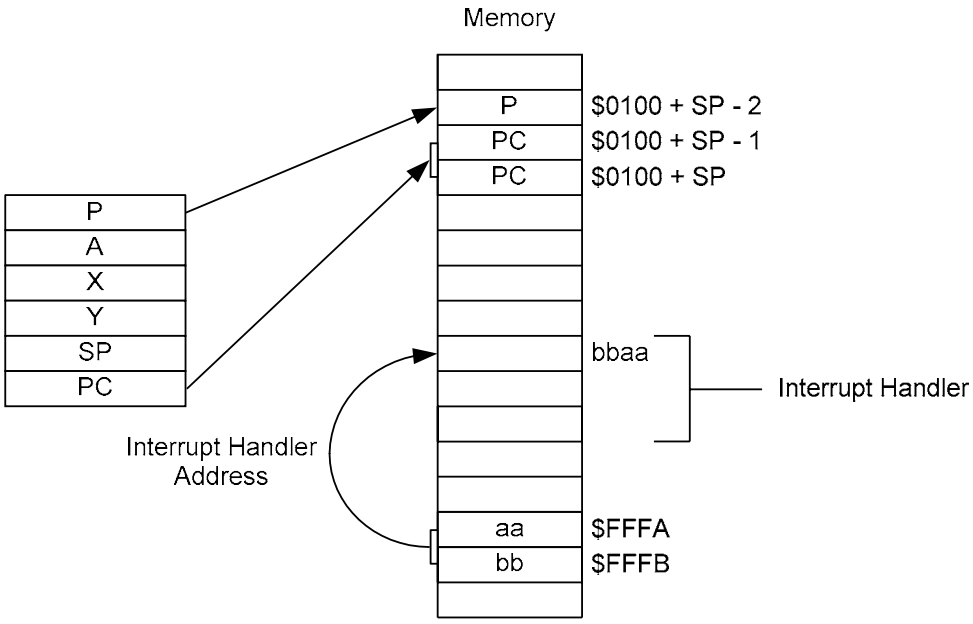


図2-5.NMI（Non-Maskable Interrupt）の処理。

2.5 アドレッシングモード

6502にはいくつかのアドレッシングモードがあり、メモリロケーションへのアクセス方法が異なります。また、メモリではなく、レジスタの内容を操作するアドレッシングモードもあります。6502には、全部で13種類のアドレッシングモードがあります。一部の命令では、複数の異なるアドレッシング・モードを使用できます。使用可能なアドレスモードの詳細については、付録Eを参照してください。

2.6 使い方

6502には56種類の命令がありますが、中には異なるアドレッシング・モードを使用した複数のバリエーションがあり、有効なオペコード（オペレーション・コード）は256種類のうち151種類です。命令セットの詳細については、[2]、[29]、[32]を参照してください。命令は、アドレッシング・モードに応じて、1バイト、2バイト、3バイトのいずれかです。最初のバイトはオペコードで、残りのバイトはオペランドです。命令はいくつかの機能グループに分類されます。

- ロード／ストア操作 - メモリからレジスタをロードしたり、レジスタの内容をメモリにストアしたりすることができます。
- レジスタ転送操作 - XまたはYレジスタの内容をアキュムレータにコピーしたり、アキュムレータの内容をXまたはYレジスタにコピーしたりします。
- スタック操作 - X レジスタを使って、スタックのプッシュやプル、スタックポインタの操作を行うことができます。
- 論理演算 - アキュムレータとメモリに格納されている値に対して論理演算を行います。
- 算術演算 - レジスタやメモリに対して算術演算を行います。
- インクリメント／デクリメント - X、Yレジスタまたはメモリに保存されている値をインクリメントまたはデクリメントします。
- シフト - アキュムレータまたはメモリロケーションのビットを左または右に1ビットずつシフトします。

- Jumps / Calls - シーケンシャルな実行順序を中断し、指定されたアドレスから再開する。
- 分岐 - ある条件が満たされた場合に、指定されたアドレスから再開して、連続した実行シーケンスを中断すること。その条件とは、ステータスレジスタの特定のビットを調べることです。
- ステータス・レジスタ操作 - ステータス・レジスタのフラグを設定またはクリアします。
- システム機能 - 使用頻度の低い機能を実行します。

3 - 画像処理ユニット

3.1 2C02の概要

また、リコーはPPUとして2C02を供給しました。PPUのレジスタは、付録にあるように、主にCPUメモリのI/Oレジスタ部の$2000-$2007と$4014に配置されています。
B.さらに、画面のスクロールに使われる特別なレジスタもあります。

3.2 PPUメモリーマップ

PPUは、VRAM（Video RAM）と呼ばれる独自のメモリを持っています。CPUと同様に、PPUも64KBのメモリをアドレス指定できますが、物理的なRAMは16KBしかありません。PPUのメモリマップを図3-1に示します。ここでも、左側のマップは簡略化されたバージョンを示しており、右側のマップで詳しく説明されています。物理アドレス空間と論理アドレス空間の違いにより、3FFF以上のアドレスは折り返されるため、論理メモリの位置$4000～$FFFFは、実質的に$0000～$3FFFのミラーとなります。

PPUのメモリーからの読み出しと書き込みは、I/Oレジスタ$2006と
2007ドルをCPUメモリに格納します。これは、画面の描画時に使用されるアドレスに影響を与えるため、表示される内容が破損する可能性があるため、通常はフレームの最後のV-Blank時に行われます。しかし、この効果を利用して分割画面の効果を出すことができます。

PPUのメモリは16ビットアドレスですが、I/Oレジスタは8ビットなので、2回の書き込みで2006ドルで必要なアドレスを設定することができます。その後、データの読み出しや書き込みが可能になります。
$2007.2007への書き込みが終わると，$2000のビット2に応じてアドレスが1または32だけ増加します。2007からの最初の読み出しは無効で、データは実際にバッファリングされ、次の読み出しで返されます。これは、カラーパレットには適用されません。

PPUは、スプライトの属性を保存するために、256バイトの独立したメモリ領域「SPR-RAM（Sprite RAM）」を持っています。スプライト自体は、パターンテーブルの中にあります。

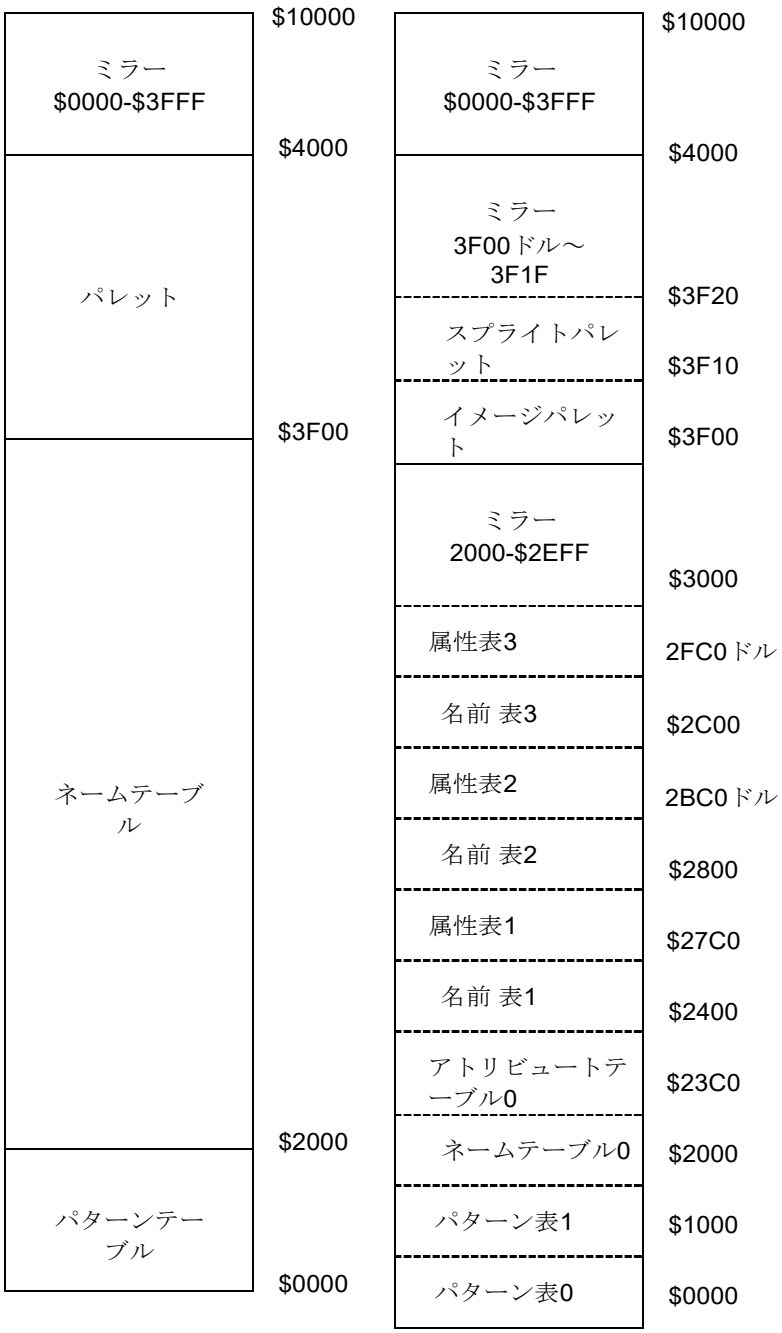


図3-1.PPUのメモリーマップ

。

3.3 PPUレジスター

CPUと他のデバイスとの通信は、メモリマップされたI/Oレジスタを介して行われます。PPUが使用するレジスタは、メインメモリの2000～2007ドルに配置されており、さらにダイレクトメモリアクセス用のレジスタが4014ドルに配置されています。なお、このレジスタは
2000-$2007は、$2008-$3FFFの領域で8バイトごとにミラーリングされます。すべてのI/Oレジスタの概要は、付録Bに記載されています。

PPUの動作は、PPU制御レジスタ1とPPU制御レジスタ2と呼ばれる$2000と$2001に書き

込むことで、CPUから制御することができます。両レジスターは

にのみ書き込まれます。£2000のビット7は、NMIを無効にするために使用することができます。この種の割り込みは、V-Blankが発生するたびに発生し、ステータスレジスタの割り込み禁止フラグの影響を受けないことを覚えておいてください。このビットをクリアすると、V-Blank時にNMIが発生しなくなります。NESは8x8と8x16の両方のスプライトをサポートしているので、$2000のビット5を設定すると、8x16のスプライトに切り替わります。PPUのメモリで次に読み書きするアドレスは、I/Oが発生するたびにインクリメントされます。インクリメントする値は、$2000のビット2の値を設定することで調整します。このビットがクリアされている場合、アドレスは1ずつ増加し（水平方向）、そうでない場合は32ずつ増加します（垂直方向）。2001を使用して、ビット3をクリアすることで背景を隠し、同様にビット4をクリアすることでスプライトを隠すことができます。

PPUステータス・レジスタは、$2002に配置されており、読み取り専用です。このレジスタは、PPUがそのステータスをCPUに報告するために使用されます。プログラムは、PPUのステータスを確認するために、頻繁にCPUにこの位置からの読み出しを行わせます。ビット7は、V-Blankが発生していることを示すためにPPUによって設定されます。ビット6とビット7はスプライトに関連しており、後述します。ビット4は、PPUがVRAMへの書き込みを受け入れるかどうかを示し、クリアなら書き込みは無視されます。2002からの読み出しが発生すると、ビット7は、$2005と$2006と同様に0にリセットされます。

3.3.1 ダイレクトメモリアクセス

大量のデータを機器間で転送する場合、プロセッサを介して転送するのは効率が悪い。例えば、CPUのメモリからスプライトのメモリにデータを転送するには、次のような手順を踏みます。

1. 必要なSPR-RAMのアドレスをCPUにロードします。
2. 必要なSPR-RAMのアドレスを$2003に書き込みます。
3. バイトをCPUにロードする。
4. バイトを$2004に書き込む。

これを256回繰り返すと、スプライトメモリの容量が増える。ダイレクト・メモリー・アクセス（DMA）は、CPUのメモリーからスプライト・メモリーへのデータのコピーをより効率的に行うための技術です。DMAを使えば、$4014への書き込みという1つの命令で、スプライト・メモリ全体を埋めることができます。CPUメモリの開始アドレスは、書き込みのオペランドに$100を掛けたもので指定されます。このアドレスから始まる256バイトが、CPUを介さずに直接スプライト・メモリにコピーされます。

DMAが発生しているときは、メモリバスが使用されているため、CPUがメモリにアクセスすることができず、その結果、それ以上の命令にアクセスすることができなくなります。これを「サイクルステイル」といい、CPUはDMAの転送が完了するまで待たなければならない。ファミコンの場合、DMAにかかる時間は512サイクル（約4.5スキャンライン分）に相当し、その後、CPUが再開できる。これは、CPUを使って手動でコピーする場合に比べてかなり少ない。

3.4 カラーパレット

ファミコンのカラーパレットは52色ですが、実際には64色分のスペースがあります。しかし、一度にすべての色を表示できるわけではありません。ファミコンでは、イメージパレット（$3F00～$3F0F）とスプライトパレット（$3F10～$3F1F）の2つのパレットがあり、それぞれ16個のエントリーがあります。イメージパレットには、背景タイルに使用できる色が表示されます。スプライトパレットには、スプライトに使用できる色が表示されま

す。これらのパレットには、実際の色の値ではなく、システムパレット内の色のインデックスが格納されています。必要なのは64個のユニークな値だけなので、ビット6と7は無視して構いません。

3F00番台のパレットエントリは背景色で、透明度を高めるために使用されます。パレットの4バイトごとに$3F00がコピーされるようにミラーリングされています。したがって、$3F04、$3F08となります。
3F0C、3F10、3F14、3F18、3F1Cは、3F00の単なるコピーであり、総数は

各パレットの色数は16ではなく13です[5]。したがって、常に画面に表示される色の総数は、52色のうち25色となります。また、両方のパレットは$3F20-$3FFFにミラーリングされています。カラーパレットを付録Fに示す。

3.5 パターンテーブル

ファミコンには、$0000と$1000の2つのパターンテーブルがあります。パターンテーブルには、画面に描画できる8x8ピクセルのタイルが格納されています。多くのゲームでは、カートリッジのCHR-ROMにパターンテーブルが格納されていますが、CHR-ROMを持たないゲームでは、パターンテーブルにRAMを使用し、実行時にパターンテーブルを埋めていきます。パターンテーブルには、画像やスプライトのパレットエントリを識別するのに必要な4ビットのうち、最下位の2ビットが格納されています（00bはパレットエントリ0、01bは1、10bは2、11bは3）。

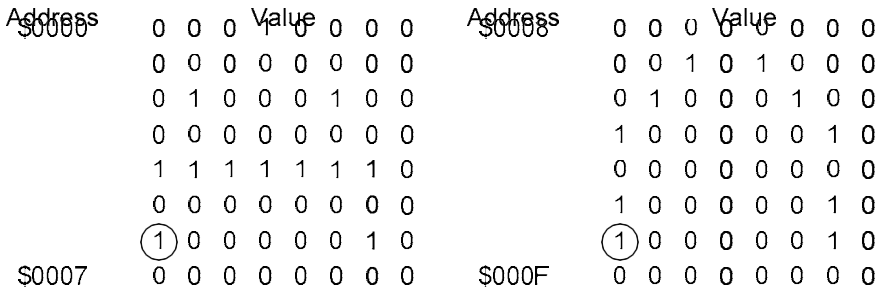


図3-2.パターンテーブル。7]より引用。

図3-2は、パターンテーブルの仕組みを示したものです。図3-2は、パターンテーブルの仕組みを示したもので、最終的には「A」という文字ができあがります（下図）。この文字は、左上から1ビット、右上から1ビットずつ取り出して2ビットの色を作り、ピクセルごとに構成されています。残りの2ビットの色は、属性テーブルから取得しています。なお、表示されている色は、ファミコン純正のカラーパレットの値ではありません。

3.6 名前のテーブル／属性のテーブル

ネームテーブルは基本的にタイル番号のマトリックスで、パターンテーブルに格納されているタイルを指しています。ネームテーブルは32x30のタイルで、各タイルは8x8ピクセルなので、ネームテーブル全体では256x240ピクセルとなります。

各名前テーブルには、関連する属性テーブルがあります。属性テーブルは、タイルの色の

上位2ビットを保持する。属性テーブルの各バイトは、タイルの4x4グループを表すので、属性テーブルはこれらのグループの8x8テーブルとなる。各4x4グループは、図3-3[9]に示すように、さらに4つの2x2の正方形に分割されます。8x8のタイルには$0-$Fの番号が付けられています。バイトのレイアウトは

は33221100で、2ビットごとに指定された正方形の最上位2色のビットを指定します。

スクエア0 $0$1 $2$3	スクエア1 45 67
スクエア2 89 AB	スクエア3 C$D EF

図3-3.4x4タイル群のレイアウト20]から引用した.

NESはネームテーブルとアトリビュートテーブルを格納するために2KBしか持っていないので、それぞれ2つずつしか格納できません。しかし、最大で4つのアドレスを持つことができます。これを可能にするのがミラーリングです。ミラーリングには4つの種類があり、論理的な名前のテーブル（アドレスを指定できるテーブル）の略語であるL1（2000ドル）、L2（2400ドル）、L3（2800ドル）、L4（2800ドル）を使って以下のように説明する。$2C00:

- 水平ミラーリングでは、図3-4に示すように、L1とL2を第1の物理名表に、L3とL4を第2の物理名表にマッピングします。

1	1
2	2

図3-4.水平方向のミラーリング。

- 垂直ミラーリングでは、図3-5に示すように、L1とL3を第1の物理名表に、L2とL4を第2の物理名表にマッピングします。

1	2
1	2

図3-5.垂直方向のミラーリング。

- シングルスクリーンミラーリングでは、図3-6に示すように、4つの論理的なネームテーブルをすべて同じ物理的なネームテーブルにポイントします。

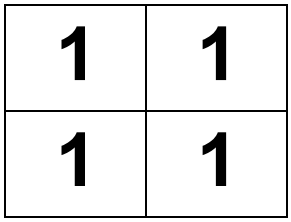


図3-6.シングルスクリーンのミラーリング。

- 4画面ミラーリングでは、カートリッジ本体に2KBのRAMを追加し、図3-7に示すように、論理的なネームテーブルをそれぞれ別の物理的なネームテーブルにマッピングできるようにしています。

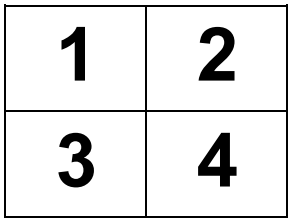


図3-7.4画面ミラーリング。

3.7 スプライト

スプライトとは、画面上に描画するキャラクターのことです。スプライトの大きさは、8×8ピクセルまたは8×16ピクセルです。ほとんどのキャラクターは複数のスプライトで構成されています。スプライトのデータはパターンテーブルに、スプライトの属性はSPR-RAMに格納されています。スプライトは最大で64個あり、それぞれがSPR-RAMの4バイトを使用する。バイトは以下のように動作する。

- バイト0 - スプライトの左上のY座標から1を引いた値を格納する。
- バイト1 - パターンテーブル内のスプライトのインデックス番号。
- バイト2 - スプライトの属性を格納します。
  - ビット0-1 - カラーの最上位2ビット。
  - ビット5 - このスプライトが背景より優先されるかどうかを示します。
  - ビット6 - スプライトを水平方向に反転させるかどうかを示します。
  - ビット7 - スプライトを垂直方向に反転させるかどうかを示します。

8x16スプライトは、インデックス番号に応じて異なるパターンテーブルを使用します。インデックス番号が偶数の場合、スプライトデータは最初のパターンテーブルの$0000に格納され、それ以外の場合は2番目のパターンテーブルの$1000に格納されます。

最初に$2003に必要なアドレスを書き込み、次に$2004を読み書きすることで、スプライトを一度に1つずつ読み書きすることができます。また、$4014に書き込むことで、1回のDMA動作でSPR-RAM全体を書き込むことができます。

スプライトは、SPR-RAM内の位置に応じて優先順位が決められています。最初のスプライトはスプライト0と呼ばれ、より高い優先度を持っています。各ラインでは、どのスプライ

トがそのラインにあるかを計算し、優先順位の低いものから順に描いていき、優先順位の高いスプライトが上に来るようにします。スキャンラインあたり8個のスプライトのみが許可されており、システムはこの数に達したときにI/Oレジスタ$2002のビット5を設定することで表示します。

スクロールによく使われる手法として、スプライト0が透明でない背景ピクセルと重なっているかどうかを判断するものがあります。システムがスプライト0を描画しているときに、その中の透明でないピクセルが透明でない背景ピクセルと同じ位置にある場合、システムは$2002のビット6にスプライト0ヒットフラグを設定します。したがって、背景タイルが透明なピクセルのみを含む場合、スプライト0ヒットフラグは設定されません。図19は、スプライト0の検出を示しています。左の画像は背景、中央の画像はスプライト、右の画像は両者の合成画像です。カラー0は透明度を表し，丸で囲んだピクセルはスプライト0ヒットフラグが設定されていることを示している。図3-8は[20]を参考にしていますが，オリジナルはヒットフラグが設定されている場所を誤って示していました。

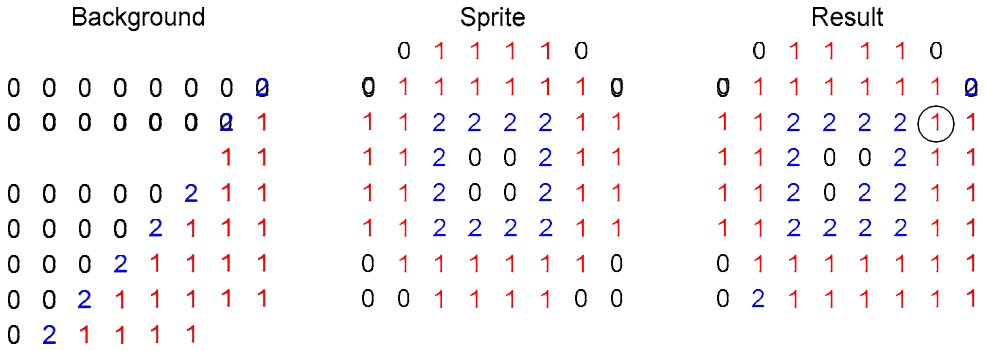


図3-8.スプライト0の検出7]より引用。

キャラクターは一般的に1つのスプライトよりも大きいので、複数のスプライトを組み合わせて作られます。例えば、図3-9はマリオのキャラクターが8×8のスプライトを8つ組み合わせて作られていることを示しています。

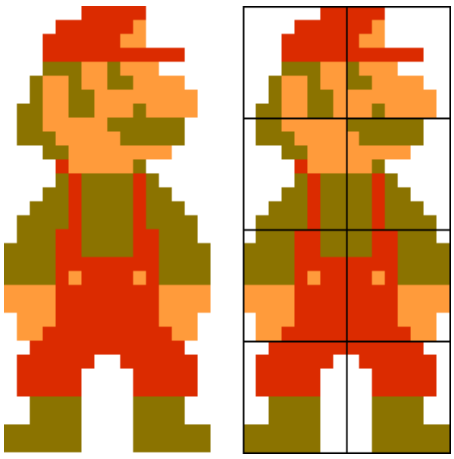


図3-9.キャラクターの構築33]からの引用です。

3.8 スクロール

背景は水平または垂直にスクロールできます。スクロールには、個別の名前テーブルが使用されます。画面上の背景は常に、1つのネームテーブルから直接取得されるか、2つのネームテーブルを組み合わせたものになります。これを図3-10と図3-11に示します。図3-10は、2つのネームテーブルの内容を示しています（もう1つは

はもちろん鏡）、図3-11は、スプライトを含めて画面に表示された合成画像です。

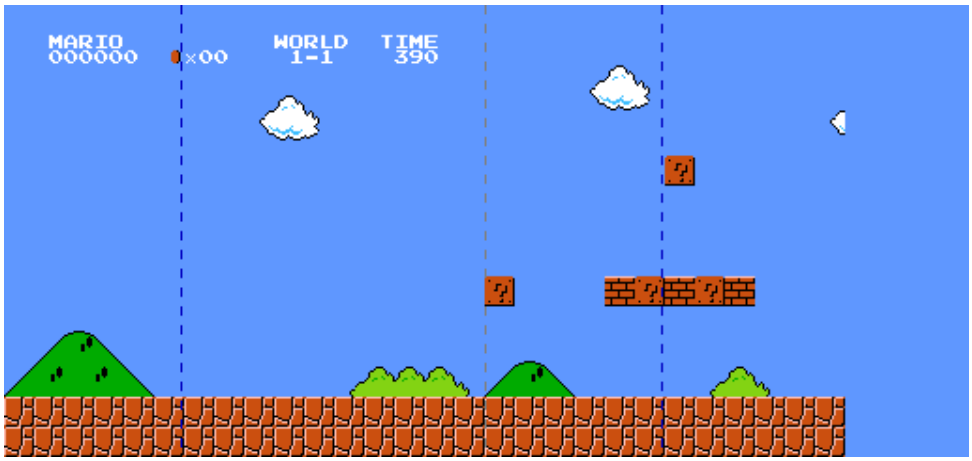


図3-10.スーパーマリオブラザーズ」の横スクロール

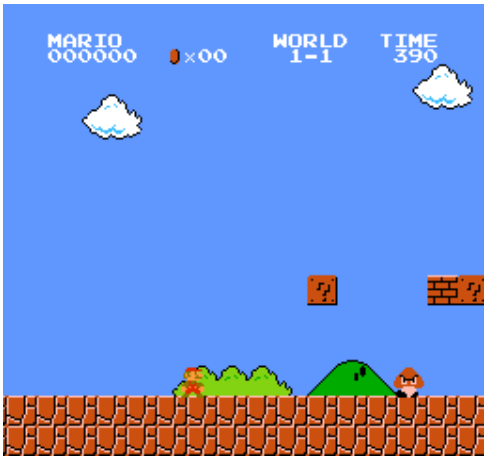


図3-11.合成画像です。

最終的なイメージは、1つ目のネームテーブルから始まり、2つ目のネームテーブルへと伸びていきます。図3-10では、2つのネーム・テーブルの間を灰色の線で示しています。2本の青い線は、画面に表示される範囲を示している。画面に表示されている部分の左側には、すでに表示されていて、画面からスクロールしている部分がある。画面の右側には、マリオが進むにつれて、システムがネームテーブルに先の情報を入力し、画面に表示しているところです。半分に切れている雲のように、この部分はまだシステムで埋め尽くされていません。ゲームによっては、一方向にしか移動できないものと、両方向にスクロールできるものがある。これについて、任天堂は次のように説明している[33]。

> 「PPUは一度に960文字しか表示できませんが、実際にはその倍の文字を記憶しています。一方通行のスクロールでは、スクロールの後ろの古い文字が常に新しい文字に置き換わります。スーパーマリオブラザーズ」などのゲームで、画面が片方向にしかスクロールしないのはそのためです。しかし、メトロイドでは、スクロールは2方向に行われ、スクロールの方向に新しいキャラクターがどんどん追加されていきます。"

画面のステータスバー領域が他と同じようにスクロールしておらず、ファーストネームテーブルに完全に常駐していることがわかる。これはステータス情報の典型で、スーパーマリオブラザーズではスプライト0のヒットフラグを使い、スーパーマリオブラザーズ3ではIRQを発生させることで対応しています。

水平スクロールと垂直スクロールの全体像を図3-12に示します。ここでAとして示されている名前のテーブルは、$2000のビット0-1で指定され、Bはそれ以降の名前のテーブルである（ミラーリングの手法に依存する）。これは、水平方向と垂直方向の同時スクロールが可能なゲームには適用されません[7]。背景画像は、図3-13のようにネームテーブルをまたいで表示されます。

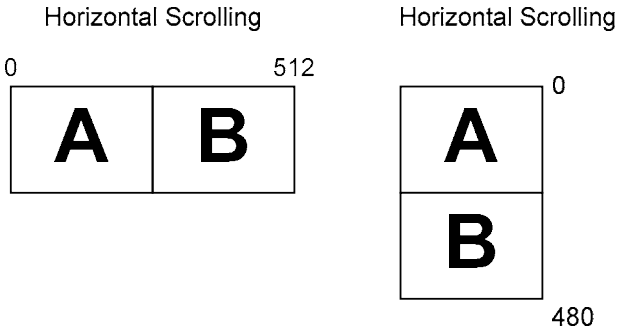


図3-12.水平方向と垂直方向のスクロール7]より引用。

ネームテーブル2 (2800円)	ネームテーブル3 ($2C00)
ネームテーブル0（2000円	ネームテーブル1 (2400ドル)

図3-13.背景に使用したネームテーブル7]からの引用です。

スクロールの仕組みは[8]に記載されていますが、ここではその概要を説明します。システムは16ビットのVRAMアドレスレジスタを保持しており、その値は$2006によって設定されます。このレジスタのレイアウトは以下の通りです。

- ビット0～11 - ネームテーブルのアドレスを2000ドルからのオフセットとして格納します。ビット0～4はx-scrollで、線が描かれるごとにインクリメントされます。31からインクリメントされると、0にラップし、ビット10が切り替わります。ビット5-9はy-scrollで、ラインの終わりにインクリメントされます。29からインクリメントされると、0にラップしてビット11が切り替わります。$2007への書き込みで値が29以上に設定されている場合、31に達したときに0に折り返されますが、ビット11は影響を受けません。
- ビット12～14は、タイルのYオフセットです。

x-scrollとy-scrollがタイルの番号を示しているので、これで画面横（256ピクセル）に32枚、画面下（240ピクセル）に30枚、合計960枚のタイルを配置することができます。

2つ目の一時的なVRAMアドレスレジスタは、16ビットの長さです。最後に、3ビットのタイルXオフセットがあります。これらは、レジスタへの書き込みやフレームの描画時に更新されます。

3.9 テレビの規格

ファミコンはテレビに接続してユーザーにゲームを見せる。そのため、NTSCとPALという2つのテレビフォーマットに合わせて、異なるバージョンのシステムが作られた。NTSC（National Television Standards Committee）は、北米、南米の大部分、アジアの一部で使用されている規格である[34]。PAL（Phase Alternating Line）は、ヨーロッパ、アジアの大部分、アフリカ、オーストラレーシアで使用されている規格である[35]。表3-1に、NTSC版とPAL版のファミコンの違いを示す。

	NTSC	PAL
フレーム/秒	60	50
1フレームあたりの時間（ミリ秒	16.67	20
スキャンライン/フレーム（うちV-ブランク	262 (20)	312 (70)
スキャンラインあたりのCPUサイクル	113.33	106.56
解像度	256 x 224	256 x 240
CPU速度	1.79MHz	1.66MHz

表3-1.NTSCとPALのNESシステムの比較。

テレビ画面に画像を表示するには、高速の電子の流れが画面を左から右に移動し、各ピクセルを描画します。ピクセルの1つのラインは、スキャンラインと呼ばれます。スキャンラインが終わると、電子ビームは次のラインに移動し、左に戻らないと先に進めない。これにかかる時間を水平ブランク期間（H-Blank）という。

画面を一度描画した後、電子ビームは左上に戻り、次のフレームを開始する準備をしなければならない。これにかかる時間をV-ブランク期間（Vertical Blank period）といいます。V-ブランク期間に入ると、PPUはI/Oレジスタ$2002のビット7をセットすることでこれを示します。このビットは、CPUが次に$2002から読み出すときにリセットされます。

NTSC版のファミコンでは、画面に240スキャンライン（ただし、上下8ラインはカットされている）があり、Vブランクに入るにはさらに3スキャンライン分のCPUサイクルが必要となる。V-ブランク期間は、次のフレームが描画されるまでに、さらに20スキャンライン分かかります。

4 - ゲームハードウェア

4.1 カートリッジ

ファミコンのゲームは、「ゲームパック」と呼ばれるカートリッジに入っていました。ゲーム本体は、カートリッジ内のROMチップに格納されている。一部のカートリッジには、ゲームを保存するために電池で動くRAMが搭載されていた。

図4-1.ファミコンの「イース」カートリッジとファミコンの「スーパーマリオブラザーズ」「ダックハント」カートリッジの比較[28ダックハント』のファミコン用カートリッジとの比較[28]。

図4-1は、ファミコンのカートリッジの違いを示したものである。ファミコンでは、任天堂が図4-1の上段に示すような基本的なカートリッジを設計したが、他のデベロッパーは様々な形、大きさ、色のカートリッジを独自に設計していた。ファミコンでは、任天堂が図4-1の下のような標準的なデザインのカートリッジを制作した。ファミコンのカートリッジは大きくなったが、その多くは無駄なスペースである。ファミコンのカートリッジは60ピンの接続で、ファミコンのカートリッジは72ピンの接続だったため、アダプターなしでは2つのフォーマットに互換性がない[28]。図4-2は、ファミコンの内部を底面から見たものです。

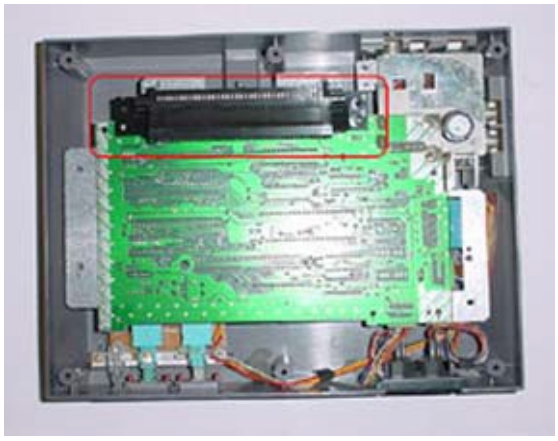


図4-2.NESの内部では、72ピンのコネクターが赤線で示されている[36]。

マザーボードです。赤い線は、カートリッジが接続される72ピンのコネクター。

図4-3は、フロントローディング方式の初代ファミコンで使用されていたカートリッジです。図4-4は、ファミコンカートリッジの内部を示したものである。左側のチップはCHR-

ROMで、ゲームのグラフィックデータであるパターンテーブルが入っている。右側のチップはPRG-ROMで、ゲームのプログラムコードが入っている。

図4-3.ファミコンに挿入されたカートリッジ[37]。

図4-4.ファミコンカートリッジの内部[38]。

4.1.1 メモリーマッパー

初期のゲームではファミコンの限られたメモリで十分だったが、ゲームが複雑になるにつれてゲームのサイズが大きくなり、メモリが足りなくなってきた。カートリッジに多くのROMを搭載するためには、必要なときにデータを出し入れできるようにしなければならない。ファミコンでは、$FFFFを超えるアドレスを指定することができないため、カートリッジ自体にあるスイッチ用のハードウェアを使用した。このハードウェアは、「メモリーマッパー」または「MMC（Memory Management Chip）」と呼ばれた。

メモリマッピングの基本的な考え方は、現在メモリに読み込まれていないROMバンクのデータにアクセスする必要が生じた場合、ソフトウェアがバンク切り替えの必要性を示し、選択されたバンクがメモリのページに読み込まれ、既存のコンテンツと入れ替わるというものである。このメモリーマッピングを採用したことが、技術的な欠陥を乗り越えてファミコンが長持ちした要因の一つである。

NESではいくつかのメモリーマッパーが使用されており、その一覧は付録Cに記載されています。一般的なメモリーマッパーを以下に紹介し、その仕組みについては付録Dに詳しい説明があります。

- UNROMスイッチは、ファミコンゲームのバンク切り替えを可能にした最初のチップである。UNROMはPRG-ROMのバンク切り替えのみ可能。CHR-ROMには対応していない。UNROMを使った16KBのPRG-ROMバンクの最大数は8である[39]。
- CNROMのスイッチでCHR-ROMのバンクを入れ替えることができたが、PRG-ROMは入れ替えることができなかった。そのため、プログラムコードの大きさはメモリーマッパーを使わないゲームと変わらないが、より高度なグラフィックが可能となった。
- MMC1では、PRG-ROMとCHR-ROMの両方のバンクを切り替えることができました。MMC1は、PRG-ROMとCHR-ROMの両方のバンクを切り替えることができ、ネームテーブルのミラーリングの変更や、RAMチップへの保存にも対応していました。MMC1は、『メトロイド』や『ゼルダの伝説』をはじめとするさまざまなゲームで使用され、最も多く使用されたメモリーマッパーでした[27]。
- MMC3では、PRG-ROMとCHR-ROMの両方のバンクを切り替えることができました。また、このチップは、画面の一部を動かし、一部を静止させるという選択的な画面スクロールを可能にし、IRQを生成することもできました。MMC3を使用した16KB PRG-ROMバンクの最大数は32です[27]。MMC3は、「スーパーマリオブラザーズ2」や「スーパーマリオブラザーズ3」などのゲームで使用され、2番目に多く使用されたチップです。

4.1.2 カートリッジのファイル形式

エミュレータを使って実行できるソフトウェアは、そのソフトウェアを格納するために使用されたオリジナルのROMチップにちなんで、通常、ROMイメージと呼ばれます。カートリッジの内容を単純にダンプしただけでは、ファイルの各部分が何を意味しているのかを識別する方法がないため、十分な情報を得ることはできません。この情報を提供するために、2つの異なるファイル形式が登場した。

iNESファイルフォーマットは、Marat Fayzullin氏が自身のiNESエミュレータで使用するために定義したものです。それ以降、ほとんどのエミュレータで使用されており、ROMイメージの最も一般的なフォーマットとなっています。INESフォーマットのファイルは、ファイル拡張子が*.nesとなっています。このフォーマットでは、ファイルの先頭に重要な情報を含む16バイトのヘッダーが用意されています。9]に記載されているフォーマットは、表4-1のようになっています。

開始バイト	長さ（Bytes	コンテンツ
0	3	iNESファイルであることを示す文字列「NES」を含む必要があります。
3	1	ファイル形式の識別にも使用されます。
4	1	16KBのPRG-ROMバンクの数。PRG-ROM（プログラムROM）とは、ROMの中で プログラムコード
5	1	8KB CHR-ROM / VROM バンクの数。CHR-ROM(Character ROM)とVROMという名称は同義語で、以下を格納するためのROMの領域を意味します。 グラフィックス情報、パターンテーブル。
6	1	ROM コントロールバイト 1: • ビット0 - ゲームで使用されるミラーリングのタイプを示し、0は水平ミラーリング、1は垂直ミラーリングを示します。 • ビット1：メモリロケーション$6000-$7FFFにバッテリーバックアップされたRAMが存在することを示します。 • Bit 2 - メモリロケーション$7000-$71FFに512バイトのトレーナーが存在することを示します。 • ビット3 - このビットがセットされると、ビット0を上書きして、4つのスクリーンミラーリングが使用されるべきであることを示します。 • ビット4-7 - マッパー番号の下位4ビット。
7	1	ROM コントロールバイト 2: • ビット0-3 - 将来の使用のために予約されており、すべて0でなければなりません。 • ビット4-7 - マッパー番号の上位4ビット。
8	1	8KB RAMバンクの数。前作との互換性のため iNESフォーマットのバージョンでは、1ページのRAMを想定していますが、これは0です。
9	7	将来の使用のために予約されており、すべて0でなければなりません。

表4-1. iNESヘッダー情報

ヘッダーに続いて、512バイトのトレーナーがある場合は、ここからROMバンクが始まり、PRG-ROM、CHR-ROMの順になっています。このフォーマットでは、最大256種類のメモリーマッパーを使用することができます。各マッパーには固有の番号が割り当てられており、マッパー番号は制御バイト2のビット4～7を4ビット左にシフトし、制御バイト1のビット4～7を加算することで得られます。マッパーとその公式iNESマッパー番号の完全なリストは、付録Cにあります。

iNESのフォーマットには多くの問題があります。例えば、ヘッダーに自分の名前を入れるなど、誤用が多いのです。最近はMarat Fayzullin氏のファミコン開発への関与が減っているようで、公式にフォーマットが更新されていないため、多くの開発者が独自に変更を加えたり、独自のマッパー番号を考案したりしています。そのため、フォーマットが不正確になり、UNIF（Universal NES Interchange Format）が開発されることになった[40]。

UNIFフォーマットのファイルは、一般的に拡張子が*.unfで、フォーマットとリビジョン番号を示すヘッダーと、それに続く一連のチャンクで構成されています。各チャンクは、そのチャンクの目的を示すID文字列、ブロックの長さ（バイト）、データで構成されています。フォーマットはXMLとよく似ていますが、XMLではタグが閉じられているのに対し、チャンクは閉じられていません。

UNIFフォーマットでは、番号ではなく、使用しているボードの名前から各マッパーを識別します。これにより、純正ボードのみを使用することができます。UNIFフォーマットはiNESフォーマットに比べて大幅に改善されていますが、現在のところ対応しているエミュレータの数が少なく、利用できるROMファイルの数も少なくなっています。今後数年のうちに、iNESフォーマットはUNIFフォーマットに徐々に取って代わられることになるでしょう。

4.2 ファミコンのディスクシステム

チップの価格上昇に対応するため、またファミコンをよりコンピュータに近づける努力の一環として、任天堂は1986年初頭に「ファミコン・ディスクシステム」を発売した[28]。このシステムは、従来のカートリッジではなく、32KBのRAMと8KBのVRAMを搭載した2.5インチの磁気ディスクに保存されたゲームをファミコンで動作させることができるものだった[9]。図4-5はファミコンに取り付けられたファミコン・ディスク・システム、図4-6はマリオ・ゴルフのディスクである。任天堂は、このシステムによって、容量が大きくなることでより大きなゲームができるようになり、また、消費者にとってはより安い価格で提供できるようになることを期待していた。また、ディスクは再利用可能なので、ゲームを交換して楽しむことができる。

図4-5.ファミコンディスクシステムに取り付けられたファミコン[28]。

ディスクシステムは、新しいディスクを購入するのではなく、専用のキオスクで少額の手数料を払って購入するものである[10]。1986年には200万台近くのディスクシステムが販売された。しかし、どのフォーマットでゲームを発売するかを決めなければならないライセンシーには不評で、また、任天堂がディスクシステムのゲームに厳しいライセンスを与えていたことも、このフォーマットを不人気にしていた。半導体の価格が下がると、カートリッジは同じ価格でディスクよりも高い容量を持つことができるようになった。ディスクシステムは1990年までに400万台以上が販売されたが、ゲームの保存にはカートリッジが主流で、ファミコンのディスクシステムはアジア以外では発売されなかった。ファミコン・ディスクシステムの仕組みについては、[9]に詳しい。

図4-6.マリオゴルフのディスク[41]。

4.3 ゲームジニー

Game Genieは、コードの実行方法を調整することでゲーマーがチート行為を行うことができる装置だった。Game Genieは、Codemasters社が設計し、Galoob Toys社が販売していた[14]。その他のチート装置は、特定のメモリ位置の値をロックすることで機能する。例えば、ゲームが残りのライフ数を$1000の位置に保存している場合、これを5にロックすれば、ゲーマーは無限のライフを手に入れることができる。しかし、Game Genieは、RAMではなくROMで動作します。カートリッジ・ポートのアドレス・バスを監視し、指定されたアドレスを検出すると、データ・バスに必要な値を書き込みます[5]。

図4-7.Game Genie [42]。

5 - 入力デバイス

5.1 コントロールパッド

6502では、メモリマップドI/O（Input/Output）を採用した。これは、I/Oデバイスとの通信にメモリと同じ命令とバスを使用し、特定のメモリ位置に書き込むと、適切なデバイスに書き込まれるというものだ。NESでは、入力デバイス用のI/Oポートは4016ドルと4017ドルだった（付録B参照）。

初代ファミコンでは、図5-1のような長方形の操作パッドを採用していた。このパッドには、A、B、スタート、セレクトの4つのボタンと、動きを制御する4方向の十字キーが付いていた。スローモーションやターボファイヤーなどの機能を追加したものなど、さまざまなバリエーションが発売されたが、オリジナルのデザインが圧倒的に多く使われていた。

図5-1.初代ファミコンのコントロールパッド[43]。

システムは、I/Oポートから複数回読み込んで、コントローラに関するすべての情報を取得します。最初の8回の読み取りはそれぞれ、A、B、Select、Start、Up、Down、Left、Rightの順に、標準コントローラの1つのボタンの状態を示します。最初のコントローラは、ポート
4016ドル、2つ目は4017ドルです。4人用のアダプターを使えば、4つのコントローラーを接続することができるが、これはまれである。この場合、コントローラ1と3は$4016に、2と4は$4017に接続されました。次の8回の読み出しは、ポート上の2番目のコントローラのステータスを取得しますが、それ以外は無視されます。

17〜20番は、デバイスが接続されているかどうか、接続されている場合はどのようなタイプのデバイスかを識別するためのシグネチャを取得します[7]。4016にジョイパッドが接続されている場合は01bが、4017にジョイパッドが接続されている場合は10bが返されます。サイクルが再び始まる前に、必要のない読み取りがあと4回あります。

I/Oデバイスからの読み出し処理は、ストローブ法を用いてリセットすることができます。リセットが必要な場合は、まずポートに1を書き込み、続いて0を書き込むことで示されます。

5.2 ザッパー

アメリカでファミコンが発売されたとき、任天堂は「ザッパー」と呼ばれるライトガンを同梱した。図5-2はオリジナルのザッパーだが、色は後にオレンジに変更された。照準器を使って狙いを定めれば、かなり正確な射撃ができる。ダックハント」、「ガムシャラ」、

「ワイルドガンマン」など、いくつかのゲームがザッパーをサポートしていた[44]。

図5-2.オリジナルのNES Zapperライトガン[45]。

"Zapper "は画面の光を受けて動作します。テレビのコントラストや明るさの調整が適切でないと、撮影したものが認識されないことがあります。(キャラクターはできるだけ明るく、背景部分はできるだけ暗くしてください）"

上記のザッパーの動作説明は、ライトガンのマニュアルから引用したものです（[44]）。基本的に，Zapperは，狙った場所の光の強さを測定することで動作します．システムは、トリガーが引かれたことを検知すると、スクリーン上のスプライトの周りに白いボックスを描きます。その後、Zapperは色の強度を確認し、スプライトである白い部分に向けられているのか、背景に属する暗い部分に向けられているのかを判断します。

付録A 算術と論理

A.1 ナンバリングシステム

16進法では、16進数で0～9、A～Fの数字を使用しますが、Aは
10とFは15を表しています。本書では、16進法を頻繁に使用しており、この形式で書かれた数字には、$と0xという接頭語が使われています（同じ意味で使われています）。例えば、$2F = (2 * 16) + 15 = 47 となります。

2進法は、基数2、桁数0と1の数字で、この方式もよく使われており、この形式で書かれた数字にはbという接尾辞が付きます。例えば、101111b＝32＋8＋4＋2＋1＝47となります。

A.2 バイナリーコード化された10進法（BCD

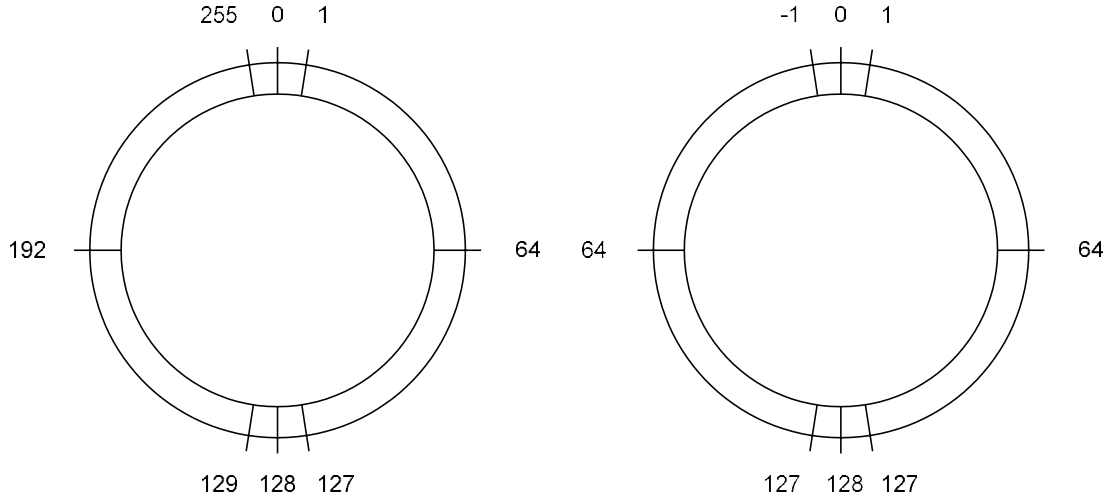
Binary Coded Decimalは、各桁を4ビットのグループで表現します。従来の2進法に比べて効率が悪いのが欠点です。例えば、123を2進法で表現すると11111011bですが、BCDで表現すると000100100011bとなります。

A.3 二人の相棒

2の補数とは、負の数を2進法で表現するための方法です。最上位のビットが符号で、0が正、1が負となります。そのため、1バイトで表現できる範囲は、0～255ではなく、-128～127となります。

A.4 ラップアラウンド

符号なしバイトの最大値は255です。同様に、0からデクリメントすると255になります。符号付きバイトの場合、正の最大値は127で、これを超えてインクリメントすると、ビット7がセットされ、値は-128となります。0付近では、正の値と負の値の間でスムーズに変化します。したがって、符号なしバイトでは、255と0の間で折り返しが発生し、符号ありバイトでは、127と-128の間で折り返しが発生します。



図A-1.符号なし（左）と符号あり（右）の8ビット整数のラップアラウンド。

付録B NES I/O
レジスター

以下の情報は[7]を参考にしています。

アドレス	アクセスレベル	説明
$2000	書く	PPUコントロールレジスタ 1: • ビット0-1 - ネームテーブルのアドレス、$2000(0)、$2400(1)、$2800(2)、$2C00(3)の4つのネームテーブルの間で変化します。 • ビット2：アドレスを増加させる量を指定し、0なら1、1なら32となります。 • ビット3 - スプライトがどのパターンテーブルに格納されているかを示します（$0000（0）または$1000（1）のいずれか）。 • ビット4 - 背景がどのパターンテーブルに保存されているかを示します（$0000（0）または$1000（1）のいずれか）。 • ビット5 - スプライトのサイズをピクセルで指定します。これが0の場合は8x8、それ以外は8x16です。 • ビット6 - PPUをマスターモードとスレーブモードの間で変更します。これはNESでは使用されません。 • ビット7 - V-Blank時にNMIを発生させるかどうかを示します。
$2001	書く	PPUコントロールレジスタ 2: • ビット0 - システムがカラーモードか（0）、モノクロモードか（1）を示す。 • ビット1 - 背景をクリップするかどうか、つまり、画面上の左8ピクセルの背景を隠すか（0）、表示するか（1）を指定します。 • ビット2 - スプライトをクリップするかどうか、つまり、画面の左8ピクセルにあるスプライトを隠すか（0）、表示するか（1）を指定します。 • ビット3 - これが0の場合、背景を表示してはいけません。 • ビット4 - これが0の場合、スプライトは表示されません。 • ビット5-7 - モノクロモードの場合は背景色、カラーモードの場合は色の濃さを示す。
$2002	リード	PPUステータスレジスタ。 • ビット4 - 設定されている場合、VRAMへの書き込みは無視されるべきであることを示します。 • ビット5 - スキャンライン・スプライト・カウント、もしセットされていれば、現在のスキャンラインに8つ以上のスプライトがあることを示します。 • ビット6 - スプライト0ヒットフラグ、スプライト0の非透過ピクセルが非透過の背景ピクセルと重なった時に設定される。

		• Bit 7 - V-Blankが発生しているかどうかを示す。
$2003	書く	SPR-RAMアドレスレジスタ。 への次の書き込みでアクセスするSPR-RAMのアドレスを保持します。 $2004.
$2004	書く	SPR-RAMのI/Oレジスタ。 2003で示されたアドレスのSPR-RAMに1バイトを書き込みます。
$2005	書く	VRAMアドレスレジスタ1。
$2006	書く	VRAMアドレスレジスタ 2.
$2007	読み取り/書き込み	VRAM I/Oレジスタ。 現在のアドレスのVRAMからバイトを読み書きします。
$4000	書く	pAPUパルス1制御レジスタ。
$4001	書く	pAPU パルス 1 ランプ制御レジスタ。
$4002	書く	pAPU Pulse 1 Fine Tune (FT) Register.
$4003	書く	pAPU Pulse 1 Coarse Tune (CT) Register.
$4004	書く	pAPUパルス2制御レジスタ。
$4005	書く	pAPU パルス 2 ランプ制御レジスタ。
$4006	書く	pAPU パルス2ファインチューンレジスタ。
$4007	書く	pAPU Pulse 2 Coarse Tune Register.
$4008	書く	pAPUトライアングル・コントロール・レジスタ 1.
$4009	書く	pAPUトライアングル・コントロール・レジスタ 2.
$400A	書く	pAPU 三角周波数レジスタ 1.
$400B	書く	pAPU 三角周波数レジスタ 2.
$400C	書く	pAPUノイズコントロールレジスタ 1.
$400E	書く	pAPUノイズ周波数レジスタ 1.
$400F	書く	pAPUノイズ周波数レジスタ 2.
$4010	書く	pAPU Delta Modulation Control Register.
$4011	書く	pAPU Delta Modulation D/A Register.
$4012	書く	pAPU Delta Modulation Address Register。
$4013	書く	pAPU Delta Modulation Data Length Registerの略。
$4014	書く	Sprite DMAレジスタ。 書き込みは、アドレス$100×n（nは書き込み値）のCPUメモリからSPR-RAMへのDMA転送を行います。
$4015	読み取り/書き込み	pAPU Sound / Vertical Clock Signal Register.

$4016	読み取り/書き込み	ジョイパッド 1: • ビット0 - ジョイパッドからデータを読み出すか、書き込み時にジョイパッドのストローブを発生させます。 • ビット3：Zapperがスプライトを指しているかどうかを示す。 • ビット4 - ザッパーのトリガーが解除されるとクリアされます。 書き込みに関わるのはビット0のみ。
$4017	読み取り/書き込み	ジョイパッド 2: 読むときは • ビット0 - ジョイパッドからデータを読み出すか、書き込み時にジョイパッドのストローブを発生させます。 • ビット3：Zapperがスプライトを指しているかどうかを示す。 • ビット4 - ザッパーのトリガーが解除されるとクリアされます。 書き込みに関わるのはビット0のみ。

付録C iNESマッパー番号

以下のマッパー番号は[9]を参考にしています。

iNESマッパー番号	マッパー名
0	NROM、マッパーなし
1	ニンテンドーMMC1
2	UNROMスイッチ
3	CNROMスイッチ
4	ニンテンドーMMC3
5	ニンテンドーMMC5
6	FFE F4xxx
7	AOROMスイッチ
8	FFE F3xxx
9	ニンテンドーMMC2
10	ニンテンドーMMC4
11	カラードリームスチップ
12	FFE F6xxx
15	100イン1スイッチ
16	バンダイチップ
17	FFE F8xxx
18	ジャレコSS8806チップ
19	ナムコット106チップ
20	任天堂ディスクシステム
21	コナミVRC4a
22	コナミVRC2a
23	コナミVRC2a
24	コナミVRC6
25	コナミVRC4b
32	アイレムG-101チップ
33	タイトー TC0190/TC0350
34	32KB ROMスイッチ
64	テンゲンRAMBO-1チップ
65	アイレムH-3001チップ
66	GNROMスイッチ
67	サンソフト3チップ
68	サンソフト4チップ
69	サンソフト5 FME-7チップ
71	カメリアチップ
78	アイレム74HC161/32ベース
91	海賊版HK-SF3チップ

付録D メモリーマッパー機能

このセクションの情報は[6]に基づいており、MMC1に関する追加情報は[46]を参照しています。

D.1 UNROMスイッチ

アドレス	データ
$8000-$FFFF	16KB PRG-ROMのバンク番号を$8000にロードする。

リセット時には、最初のPRG-ROMバンクが$8000に、最後のPRG-ROMバンクが$C000にロードされます。切り替えができるのは$8000のバンクのみで、$C000のバンクは永久にその位置に割り当てられます。このマッパーはVROMをサポートしていないので、このマッパーを使用したゲームでは、PPUメモリの$0000に8KBのVRAMが搭載されています。

D.2 CNROMスイッチ

アドレス	データ
$8000-$FFFF	8 KB CHR-ROMのバンク番号を$0000のPPUメモリにロードする。

このマッパーを使うと、PRG-ROMはNROM（マッパーなし）と同じように機能するので、PRG-ROMの16KBバンクが1つしかないゲームでは、8000ドルとC000ドルの両方にロードされ、2つあるゲームでは、1つを8000ドルに、もう1つをC000ドルにロードされます。リセット時には、最初の8KBのVROMバンクがPPU $0000にロードされます。

D.3 MMC1

アドレス	データ
8000-$9FFF	レジスタ0です。 • ビット0 - ミラーリングを水平方向（0）と垂直方向（1）で選択します。 • ビット1 - 0に設定するとシングルスクリーンミラーリングが行われます。 • ビット2 - 0の場合、PRG-ROMは$C000でスワップされます。1の場合、PRG-ROMは、$C000にスワップされます。 $8000. • ビット3 - 0の場合、PRG-ROMの32KBを$8000にスワップします。1の場合、ビット2で指定されたアドレスに16KBをスワップします。 • ビット4 - カートリッジにVROMがある場合、0はPPUの$0000で8KBのVROMをスワップすることを示し、1はPPUで2つの4KBのVROMページをスワップすることを示す。 0000と$1000です。1024KBのカートリッジでは、このビットは256KBの選択レジスタ1を使用するかどうかを指定します。 • ビット7 - 1に設定すると、レジスタがリセットされます。

$A000-$BFFF	レジスタ 1: • ビット0-3 - PPU $0000にロードするVROMバンク番号。レジスタ0のビット4に基づいて、これは8KBバンクn、または4KBバンクnと(n + 1)のいずれかになります。 • ビット4 - 256KB選択レジスタ0 1024KBカートリッジでレジスタ0のビット4がセットされている場合、256KB PRG-ROM選択の下位ビットが格納されます。それ以外の場合、0は最初の256KBのPRG-ROMからのスワップ、1は3番目の256KBのPRG-ROMからのスワップを示します。512KBの場合 カートリッジの場合、0は第1の256KBのPRG-ROMからのスワップ、1は第2の256KBのPRG-ROMからのスワップを示します。
	• ビット7 - 1に設定すると、レジスタがリセットされます。
$C000-$DFFF	レジスタ2 • ビット0-3 - PPU $1000にロードするVROMバンク番号。レジスタ0のビット4がセットされていれば、4KBのバンクnと(n + 1)となり、nはビット0〜3の値となりますが、そうでなければ無視されます。 • ビット4 - 256KB選択レジスタ1。256KB PRG- ROMセレクションの上位ビットを1024KBカートリッジに格納します。 • ビット7 - 1に設定すると、レジスタがリセットされます。
$E000-$FFFF	レジスタ 3: • ビット0-3 - メモリにロードするPRG-ROMのバンク番号。レジスタ0のビット3がクリアの場合、$8000で32KBをスワップし、そうでない場合はレジスタ0のビット2に基づいて$8000または$C000のいずれかで16KBバンクをスワップします。 • ビット7 - 1に設定すると、レジスタがリセットされます。

リセット時には、最初のPRG-ROM バンクが$8000 に、最後のPRG-ROM バンクが$C000 に読み込まれます。値は、MMC1のレジスタに1ビットずつ、5ビットが書き込まれるまで書き込まれます。ビット7 を設定して値を書き込むことで、このバッファーがリセットされ、次の書き込みはレジスタのビット0 になります。このバッファリングは、別のレジスタに書き込むことでもリセットされます。NES# のMMC1 のインプリメンテーションでは、256 KB スワッピングは現在サポートされていません。

D.4 MMC3

アドレス	データ

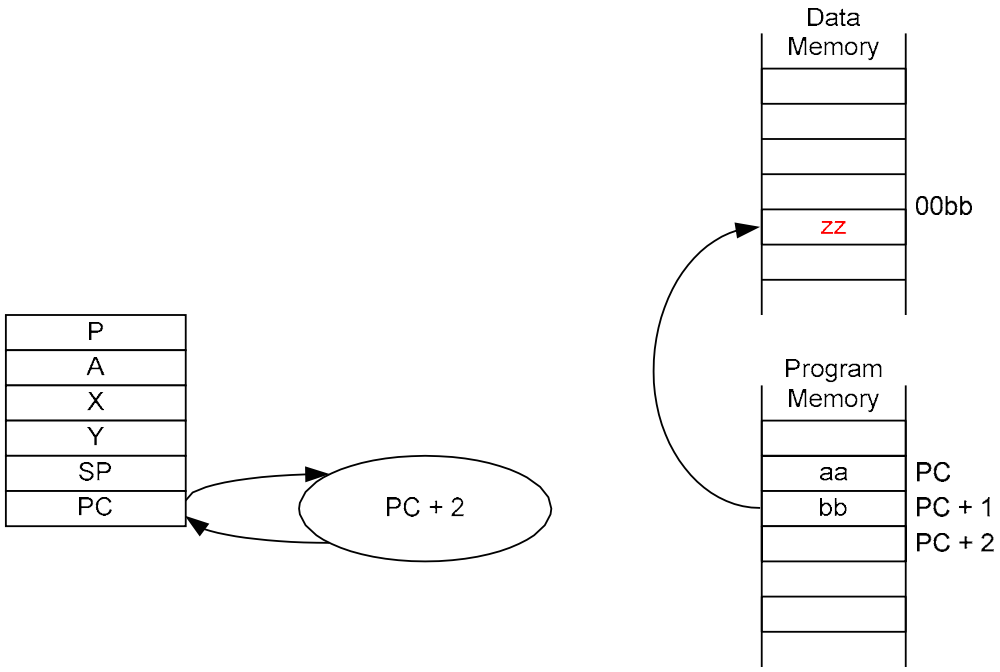
$8000	• ビット0-2 - コマンド番号。 • 0 - PPU $0000で2つの1KB VROMバンクをスワップします。 • 1 - PPU $0800で2つの1KB VROMバンクをスワップします。 • 2 - PPUの$1000で1KBのVROMバンクを1つ交換。 • 3 - PPUの1400ドルで1KBのVROMバンクを1つ交換。 • 4 - PPUの1800ドルで1KBのVROMバンクを1つ交換。 • 5 - PPU $1C00で1KBのVROMバンクを1つ交換します。 • 6 - PRG-ROMバンクをビット6に基づいて$8000または$A000のいずれかに交換します。 • 7 - PRG-ROMバンクをビット6に基づいて$A000または$C000のいずれかに交換します。 • ビット6 - 0の場合、$8000と$A000でのスワッピングが有効になり、そうでない場合は$A000と$C000でのスワッピングが有効になります。 • ビット7：1の場合、コマンド0～5のアドレスは、指定されたアドレスと$1000の排他的論理和になります。
$8001	これをページとして、$8000で指定されたコマンドを実行する の数字が表示されます。
$A000	• ビット1 - ミラーリングを水平方向（0）と垂直方向（1）で選択します。
$A001	• ビット7 - $6000-$7FFFのセーブRAMを有効にするために設定します。
$C000	IRQへのカウントダウンに使用されるIRQカウンタレジスタ。
$C001	後でIRQカウンタ・レジスタにコピーするための一時的な値を格納するために使用されるIRQラッチ・レジスタ。
$E000	IRQ制御レジスタ0は，IRQの生成を無効にし，IRQラッチレジスタをIRQカウンタレジスタにコピーするために使用する。
$E001	IRQ制御レジスタ1は、IRQ生成を有効にするために使用されます。

VROMを搭載したカートリッジでは、リセット時に最初の8KBバンクがPPU $0000にスワップされます。

付録E
6502のアドレッシングモ

E.1 ゼロページ

ゼロページアドレッシングでは、1つのオペランドを使用します。このオペランドは、操作対象のデータが存在するゼロページ（$0000～$00FF）のアドレスへのポインタとして機能します。ゼロページアドレッシングでは、オペランドに1バイトしか必要としないため、2つのオペランドを使用するアドレッシングモードに比べて、命令が短くなり、実行速度も速くなります。ゼロページ命令の例として、AND $12があります。



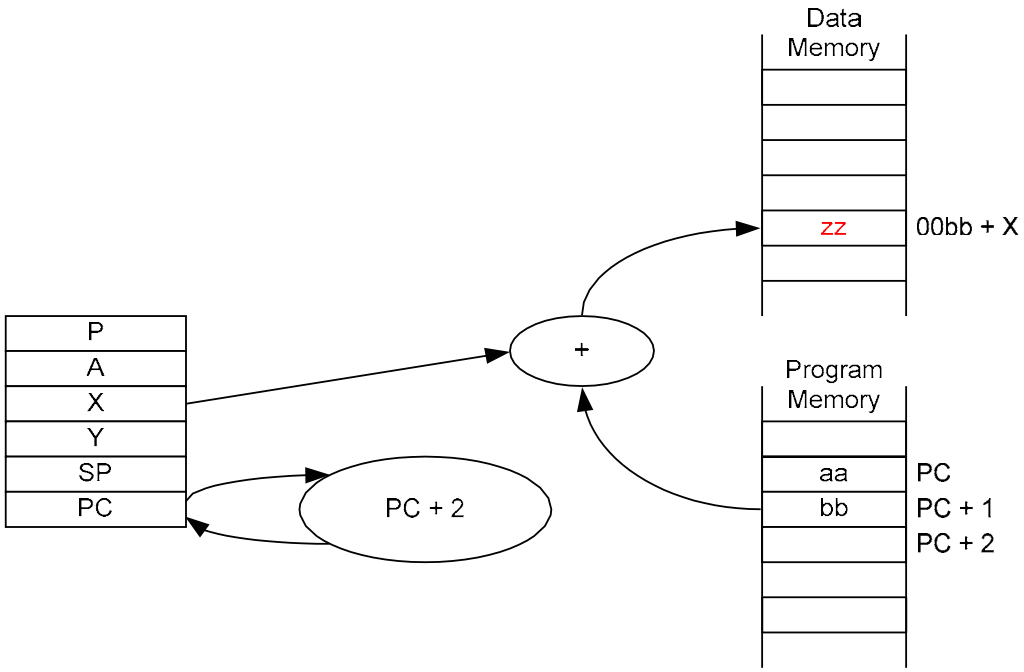
図E-1.ゼロページアドレッシング

E.2 インデックス付きゼロページ

インデックス付きゼロページアドレッシングでは、1つのオペランドにレジスタの値を加算し、データが存在するゼロページ（$0000～$00FF）のアドレスを指定します。インデックス付きゼロページアドレッシングには2つの形式があります。

- Zero Page, X - Xレジスタの内容をオペランドに追加します。これは、インデックス付きゼロページの最も一般的な形式です。このアドレッシング・モードの例は、AND $12,Xです。
- ゼロページ、Y - Yレジスタの内容をオペランドに追加します。このモードはLDX(Load X Register)とSTX(Store X Register)でのみ使用できます。このアドレッシング・モードの例は、LDX $12,Yです。

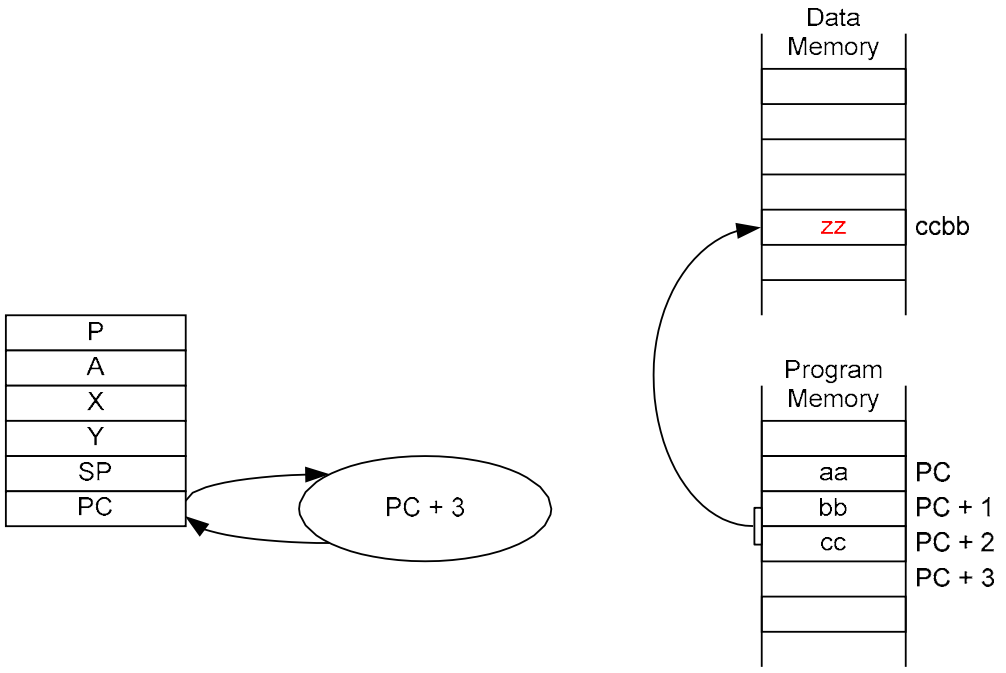
ラップアラウンドは加算時に使用されるため、データのアドレスは常にゼロページになります。例えば、オペランドが$FFで、Xレジスタが$01の場合、データのアドレスは$0100ではなく$0000になります。



図E-2.インデックス付きゼロページアドレッシング

E.3 アブソリュート

アブソリュートアドレッシングでは、演算するデータのアドレスは、供給された2つのオペランドによって指定され、最下位バイトが最初になります。絶対命令の例としては、AND $1234.

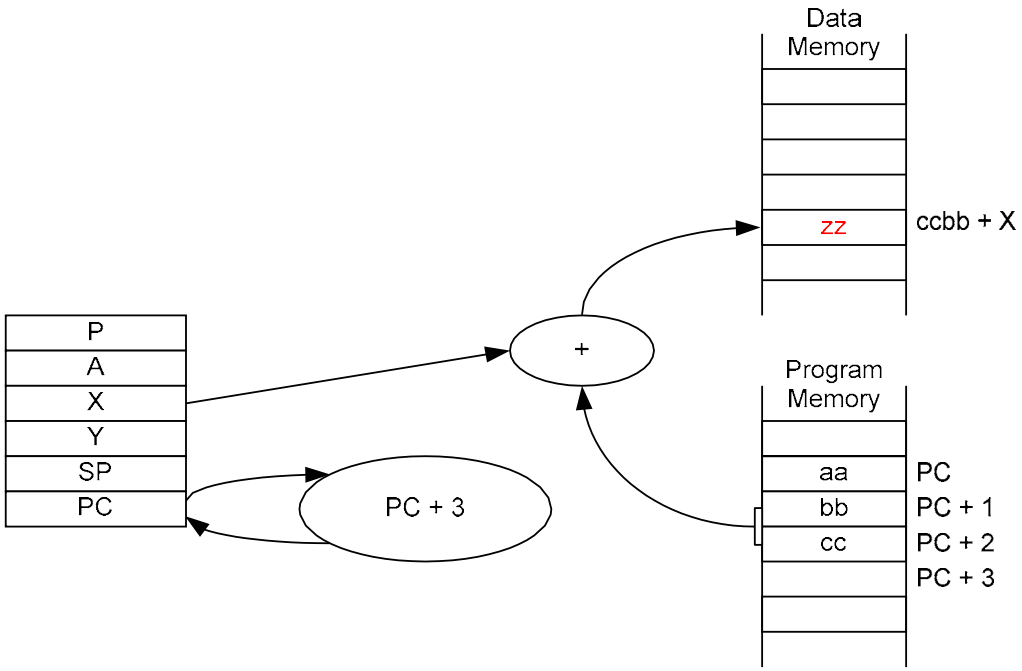


図E-3.絶対的なアドレス指定。

E.4 インデックス付きアブソリュート

インデックス付きアブソリュートアドレッシングは、2つのオペランドを最下位バイトから順に16ビットのアドレスを形成し、これにレジスタの値を加えることで、データを見つけることができるアドレスを与えます。例えば、オペランドがbbとccの場合、データのアドレスはccbb + Xとなります。 インデックス付きアブソリュートアドレッシングには2つの形式があります。

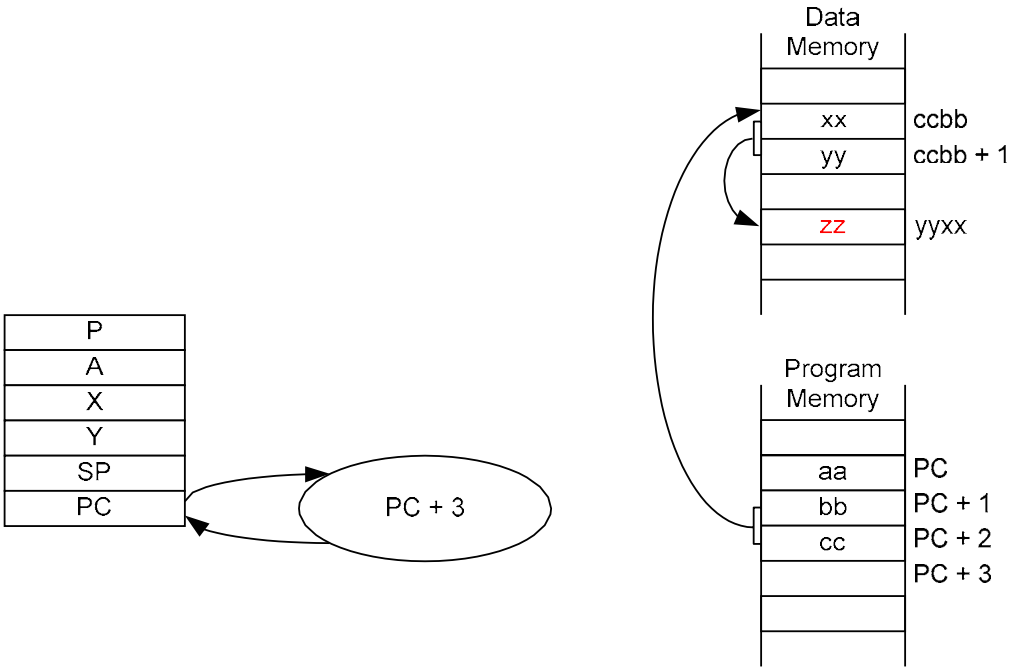
- Absolute, X - X レジスタの内容をオペランドに追加します。このアドレッシング・モードの例は、AND $1234.Xです。
- Absolute, Y - Y レジスタの内容をオペランドに追加します。このアドレッシング・モードの例は、AND $1234.Yです。



図E-4.インデックス付きアブソリュートアドレッシング。

E.5 間接的な

間接アドレスでは、2つのオペランドを用いて16ビットのアドレスを形成し、データが格納されている別のアドレスの最下位バイトを特定します。例えば、オペランドがbbとccで、ccbbにxx、ccbb+1にyyが含まれている場合、実際のターゲットアドレスはyyxxとなります。6502では、JMP（ジャンプ）のみがこのアドレッシングモードを使用し、例としてJMP（$1234）があります。図は、間接アドレスの一般的な形を示しています。しかし、JMP命令では、yyxxがデータを指し、プログラムカウンタが3増加する代わりに、プログラムカウンタがyyxxに設定され、そのアドレスから実行が再開されます。



図E-5.インダイレクト・アドレッシング

E.6 インプライド

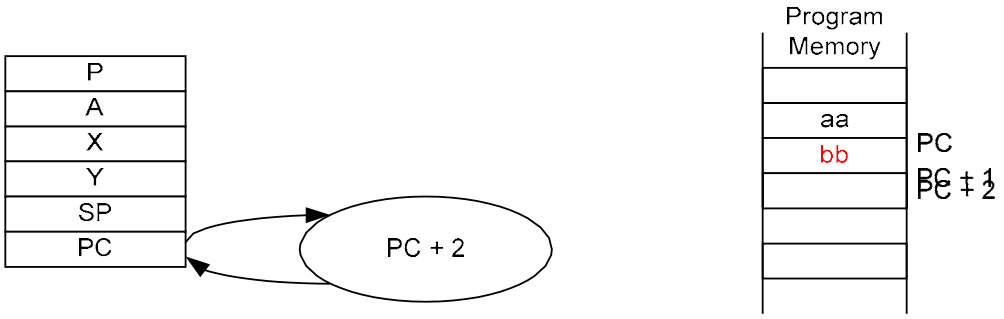
多くの命令は、メモリに格納されているオペランドへのアクセスを必要としません。CLD(Clear Decimal Mode)やNOP(No Operation)などがその例である。

E.7 アキュムレータ

アキュムレータの内容を直接操作する命令もあります。このアドレッシングモードを使用する唯一の命令は、シフト命令であるASL（Arithmetic Shift Left）、LSR（Logical Shift Right）、ROL（Rotate Left）、ROR（Rotate Right）です。

E.8 即時

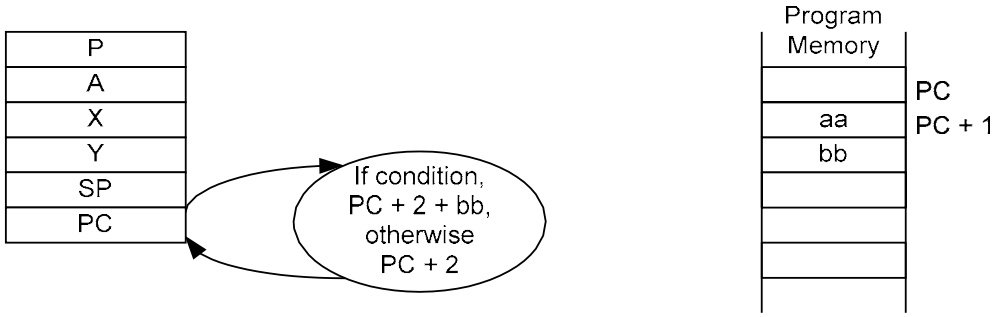
即値呼び出しを使用する命令は、命令のオペランドとして与えられた定数を直接操作します。即時命令は、AND #$12のように、オペランドの前に#を付けて表示します。



図E-6.イミディエイト・アドレッシング

E.9 相対的な

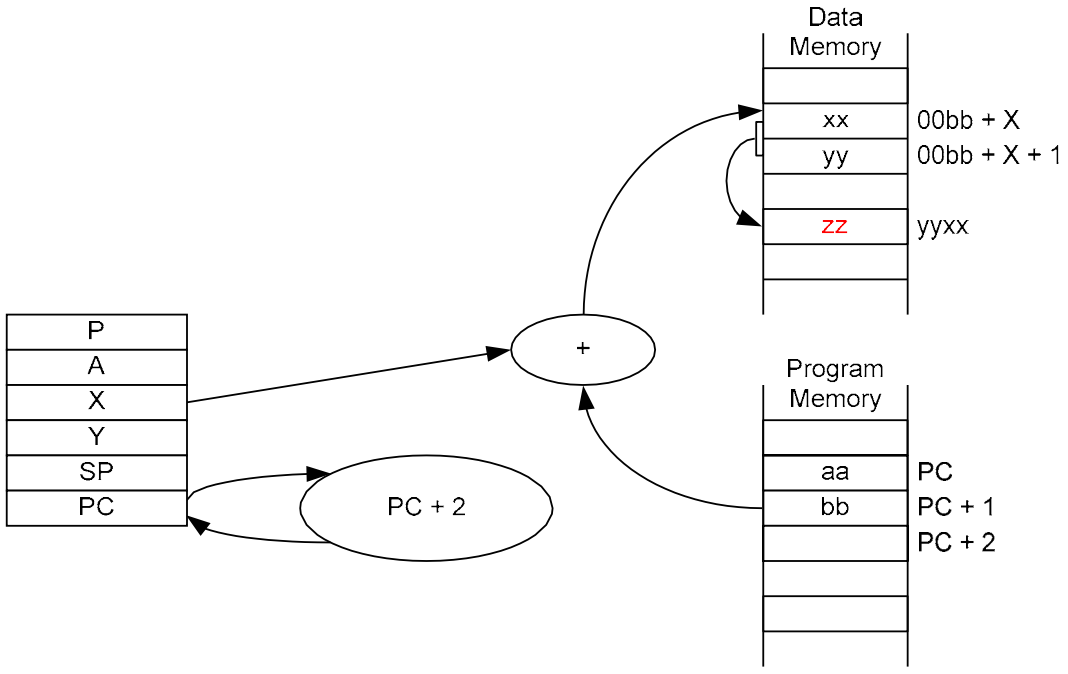
相対アドレスは、分岐命令で使用される。このアドレッシングモードでは、ある条件を満たした場合にプログラムカウンタの値が変化します。その条件は命令によって異なります。プログラムカウンタは、条件の結果にかかわらず2つインクリメントしますが、条件が真であれば、1つのオペランドがプログラムカウンタに加算され、新しい値が得られます。このため、オペランドは符号付きのバイトとして解釈され、-128～127の範囲で前方および後方への分岐が可能です。このアドレッシングモードの例は、BCS *+5で、*はプログラムカウンタの現在の値を表します。



図E-7.相対的なアドレッシング

E.10 インデクテッド・インダイレクト

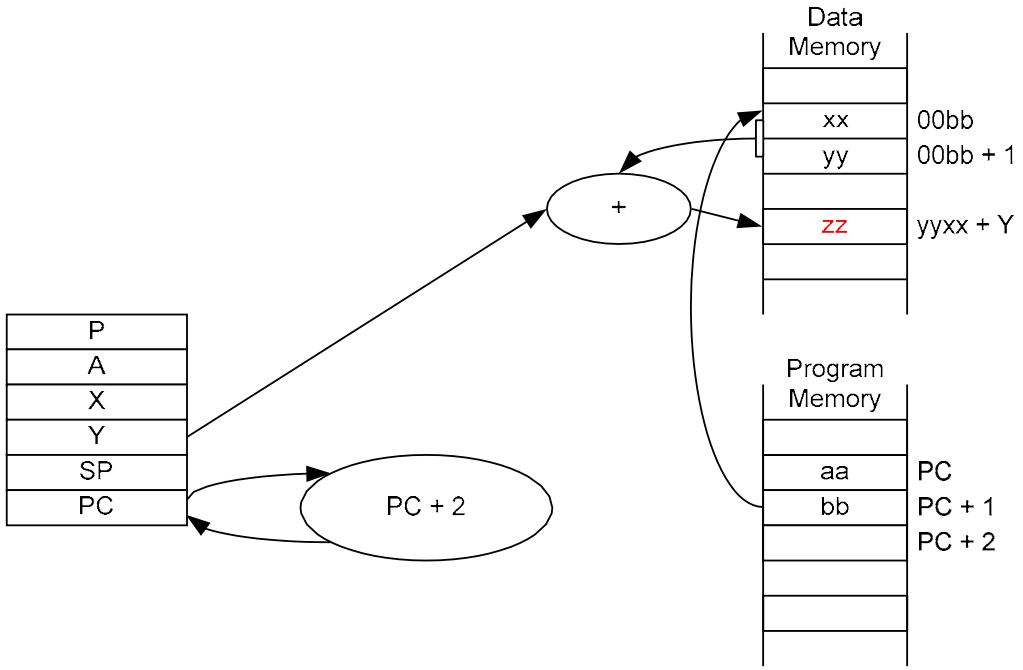
インデックス付き間接アドレス（pre-indexedとも呼ばれる）は、1バイトをオペランドとし、これにXレジスタの値を（ラップアラウンドで）加算して、ターゲットアドレスの最下位バイトのアドレスを与える。例えば、オペランドがbbの場合、00bbはxx、00bbは + 1がyyの場合、データはyyxxにあります。このアドレッシングモードの例は、AND ($12,X)です。



図E-8.インデックス付き間接アドレッシング。

E.11 インダイレクト インデックスド

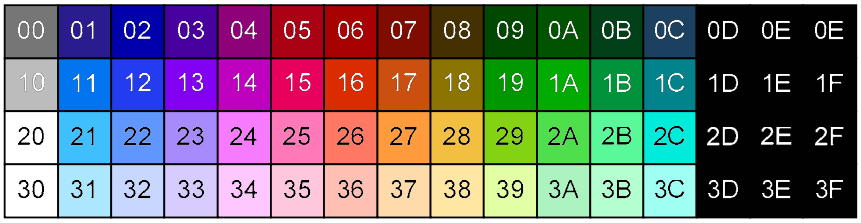
間接インデックス方式（ポストインデックス方式とも呼ばれる）のアドレッシングでは、1つのオペランドで16ビットアドレスの最下位バイトのゼロページアドレスを指定し、それをYレジスタに加算することでターゲットアドレスを得ることができます。例えば、オペランドがbb、00bbがxx、00bb+1がyyの場合、データはyyxxにあることになります。このアドレッシングモードの例は、AND ($12),Yです。



図E-19.間接的なインデックス方式のアドレッシング。

付録F NESカラーパレット

NESのカラーパレットには様々な解釈があります。47]で定義されているパレットを以下に示す。代替案は[5]と[48]に示されている。



図F-1.NESのカラーパレット。

パレットエントリー	RGB値	パレットエントリー	RGB値
00	75, 75, 75	20	FF, FF, FF
01	27, 1B, 8F	21	3F、BF、FF
02	00, 00, AB	22	5F, 97, FF
03	47, 00, 9F	23	A7、8B、FD
04	8F, 00, 77	24	F7、7B、FF
05	AB, 00, 13	25	FF, 77, B7
06	A7, 00, 00	26	FF, 77, 63
07	7F, 0B, 00	27	FF, 9B, 3B
08	43, 2F, 00	28	F3、BF、3F
09	00, 47, 00	29	83, D3, 13
0A	00, 51, 00	2A	4F、DF、4B
0B	00, 3F, 17	2B	58, F8, 98
0C	1B, 3F, 5F	2C	00、EB、DB
0D	00, 00, 00	2D	00, 00, 00
0E	00, 00, 00	2E	00, 00, 00
0F	00, 00, 00	2F	00, 00, 00
10	BC, BC, BC	30	FF, FF, FF
11	00, 73, EF	31	AB, E7, FF
12	23歳、3B、EF	32	C7、D7、FF
13	83, 00, F3	33	D7、CB、FF
14	BF, 00, BF	34	FF、C7、FF
15	E7, 00, 5B	35	FF、C7、DB
16	DB、2B、00	36	FF, BF, B3
17	CB、4F、0F	37	FF、DB、AB
18	8B, 73, 00	38	FF、E7、A3
19	00, 97, 00	39	E3、FF、A3
1A	00, AB, 00	3A	AB、F3、BF
1B	00, 93, 3B	3B	B3、FF、CF
1C	00, 83, 8B	3C	9F、FF、F3
1D	00, 00, 00	3D	00, 00, 00
1E	00, 00, 00	3E	00, 00, 00
1F	00, 00, 00	3F	00, 00, 00

リファレンス

[1] Patrick Diskin, "Nintendo Entertainment System Emulator", School of Computer Science, The University of Birmingham, 2004

[2] Andrew John Jacobs, "6502 Reference", http://www.obelisk.demon.co.uk/6502/reference.html, 2002

[3] Andrew John Jacobs, "6502 Instructions", http://www.obelisk.demon.co.uk/6502/instructions.html, 2002

[4] Andrew John Jacobs, "6502 Architecture", http://www.obelisk.demon.co.uk/6502/architecture.html, 2002

[5] Chris Covell, "NES Technical / Emulation / Development FAQ 1.4", NesDev, http://nesdev.parodius.com/NESTechFAQ.htm#nessnescompat, 2002

[6] Firebug, "Comprehensive NES Mapper Document 0.8", NesDev, http://nesdev.parodius.com/mappers.zip, 1999

[7] Jeremy Chadwick, "Nintendo Entertainment System Documentation 2.0", NesDev, http://nesdev.parodius.com/ndox200.zip, 1999.

[8] Loopy, "The Skinny on NES Scrolling", NesDev, http://nesdev.parodius.com/loopyppu.zip, 1999年

[9] Marat Fayzullin, "Nintendo Entertainment System Architecture 2.4", Department of Computer Science, University of Maryland, http://fms.komkon.org/EMUL8/NES.html (リンクはもう機能しません。バージョン2.2のコピーは http://oldnes.sourceforge.net/doc/NES- にあります) %20by%20Marat%20Fayzullin.html), 2002

[10] デビッド・シェフ、 *『Game Over』。Nintendo's Battle To Dominate An Industry*、Hodder and Stoughton、1993年

[11] 任天堂、「会社沿革」、Nintendo of America Inc、http://www.nintendo.com/corp/history.jsp、2004年

[12] ジョナサン・スミス, *Construction Complete?Computer Gaming's Battle To Take Over The World*, Future Publishing Ltd, 2000

[13] Dale Hansen, "Nintendo Entertainment System / Famicom Console Information", コンソールデータベース、http://consoledatabase.retrofaction.com/consoleinfo/nes/

[14] GameSpy、「ザ・ミュージアム」。Nintendo Entertainment System 」, GameSpy, http://www.classicgaming.com/museum/nes/

[15] Aaron Mims, The Video Game Museum, http://www.vgmuseum.com/systems/topnes/nes.jpg

[16] Marcus Liedholm、Mattias Liedholm、"The History of Nintendo Entertainment System or Famicom"、Nintendo Land、http://www.nintendoland.com/nes/history.htm

[17] Marat Fayzullin, "How To Write a Computer Emulator", Department of Computer Science, University of Maryland, http://fms.komkon.org/EMUL8/HOWTO.html (リンクはもう機能しません。コピーは http://people.ac.upc.es/vmoya/docs/HowToMarat.html にあります。)

[18] Arnold Burdett, Diana Burkhardt, John Cushion, Aline Cumming, Alan Hunter, Frank Hurvid, Thomas Ng, Tim Reeve, John Southall, Brian Jackson, John Jaworski, Graham Rogers, *The British Computer Society - A Glossary of Computing Terms (Ninth Edition)*, pp.30-31, Longman, 1998.

[19] Ian Buck, "Hardware Console Design", Department of Computer Science, Princeton University, http://graphics.stanford.edu/~ianbuck/proj/Nintendo/Nintendo.html, 1998

[20] Wikipedia, "Binary translation", Wikipedia, http://en.wikipedia.org/wiki/Binary_translation, 2003

[21] Sam Michaels, "Zophar's Domain:NES Emulators", Zophar's Domain, http://www.zophar.net/nes.html, 2004

[22] Nikolas Gavalas, "How to write an emulator", Department of Computer Engineering and Computer Science, College of Engineering, California State University Long Beach, http://www.cecs.csulb.edu/~hill/cecs497/nestreme/howto.html, 2002.
[23] TeamKNOx, "ChameleonNES", http://home.att.ne.jp/gamma/TeamKNOx/ChameleonNES/ChameleonNES.html, 2003年

[24] RealityMan, "UltraHLE - Nintendo 64 High Level Emulator", www.ultrahle.com, 2003
[25] 任天堂、"Legal Information (Copyright, Emulators, ROMs, etc.)", Nintendo of America Inc, http://www.nintendo.com/corp/legal.jsp
[26] The Scribe, "The Nintendo Emulation FAQ v2.0 - A Commentary", EmulationZone, http://www.emulationzone.org/articles/emufaq/NFAQ20_response.zip, 1999
[27] Marcus Liedholm氏とMattias Liedholm氏、「Nintendo Entertainment System (NES) or Famicom Tech specs and Hardware」、Nintendo Land、http://www.nintendoland.com/home2.htm?nes/tech.htm、2000年
[28] Christian Nutt and Benjamin Turner, "Nintendo Famicom: 20 Years Of Fun", GameSpy, http://archive.gamespy.com/articles/july03/famicom/index.shtml, 2003
[29] Lance A. Leventhal, *6502アセンブリ言語プログラミング（第2版）*, マグロウヒル社、1986年
[30] Alan Clements, *The Principles of Computer Hardware (Second Edition*), pp.363-366, オックスフォード、1991年
[31] Stuart Anderson, *Microprocessor Technology*, Newnes, 1994
[32] John Picken, "6502 Opcodes", 6502.org, http://www.6502.org/tutorials/6502opcodes.htm, 2001
[33] 任天堂、「Nintendo Power Issue 22」、NESPlayer.com、http://www.nesplayer.com/technical/tech.htm
[34] ウィキペディア「NTSC」、ウィキペディア、http://en.wikipedia.org/wiki/NTSC、2004年
[35] Wikipedia, "PAL", Wikipedia, http://en.wikipedia.org/wiki/PAL, 2004
[36] Video Game Exchange, "NES 72-pin Repair", Video Game Exchange, http://www.videogex.com/repair.htm
[37] Cory Archangel, "Game Mods", 21C Magazine, http://www.21cmagazine.com/issue2/cory_clouds.html, 2003
[38] Michael Martin-Banks, "Test Carts", NESPlayer.com, http://www.nesplayer.com/features/test%20carts/test.htm
[39] Martin Nielsen, "The Nintendo Entertainment System (NES) FAQ 3.0A", http://www.neshq.com/hardware/general/nintendo_entertainment_system.txt, NESHQ.com, 1997
[40] Tennessee Carmel-Beilleux, "UNIF File Format Specifications", NesDev, http://www.parodius.com/~veilleux/UNIF_current.txt, 2000
[41] Mikko Heinonen, Ville Heinonen and Manu Parssinen, "Nintendo Famicom Disk System", Pelikonepeijoonit, http://www.pelikonepeijoonit.net/cgi-bin/page.cgi?pkpcode=famidisk
[42] Michael Martin-Banks, "Game Genie", NESPlayer.com, http://www.nesplayer.com/database/accessories/gg.htm
[43] Michael Martin-Banks, "NES Controller (Basic)", NESPlayer.com, http://www.nesplayer.com/database/accessories/images/control.jpg
[44] Jason Lunsford, "The Zapper FAQ 0.9", NESHQ.com, http://www.neshq.com/hardware/general/nes_zapper.txt, 2000.
[45] RetroGames, http://www.retrogames.co.uk/stock/assets/images/Mach_-_Nes_Zapper.jpg
[46] Matthew J. Richey, "Nintendo MMC1 info for 8-bit NES carts", NesDev, http://nesdev.parodius.com/mmc1.txt.
[47] ルーピー、「NES Palette」、NesDev、http://nesdev.parodius.com/pal.txt
[48] Matt Conte, "NES Palette", NesDev, http://nesdev.parodius.com/nespal.txt