# ノイマンアーキテクチャの呪縛との戦い

長嶋 洋一

ASL(Art & Science Laboratory)

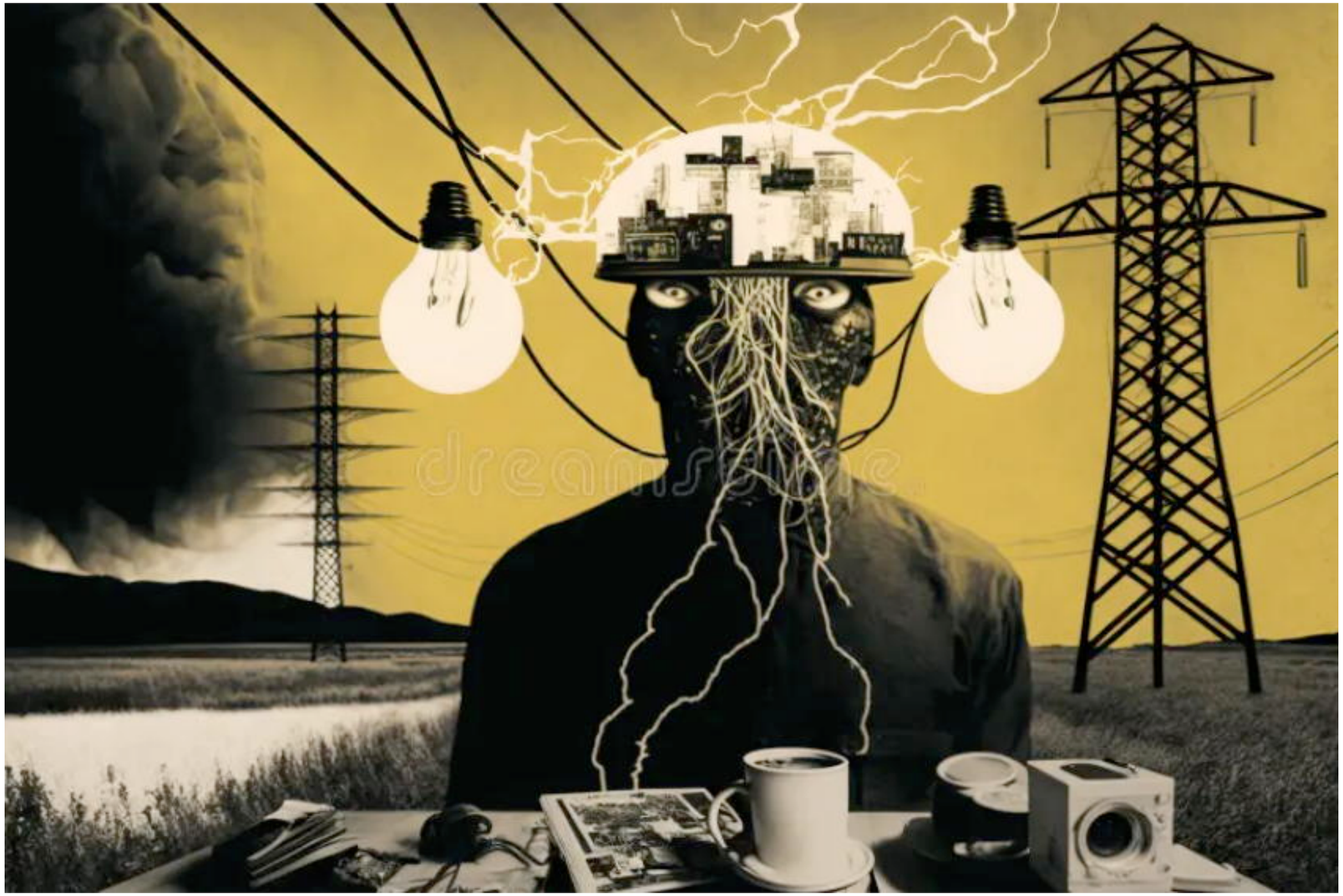
# この画像から何を読み取りますか?

ノイマンアーキテクチャの呪縛との戦い

ノイマンアーキテクチャの呪縛との戦い

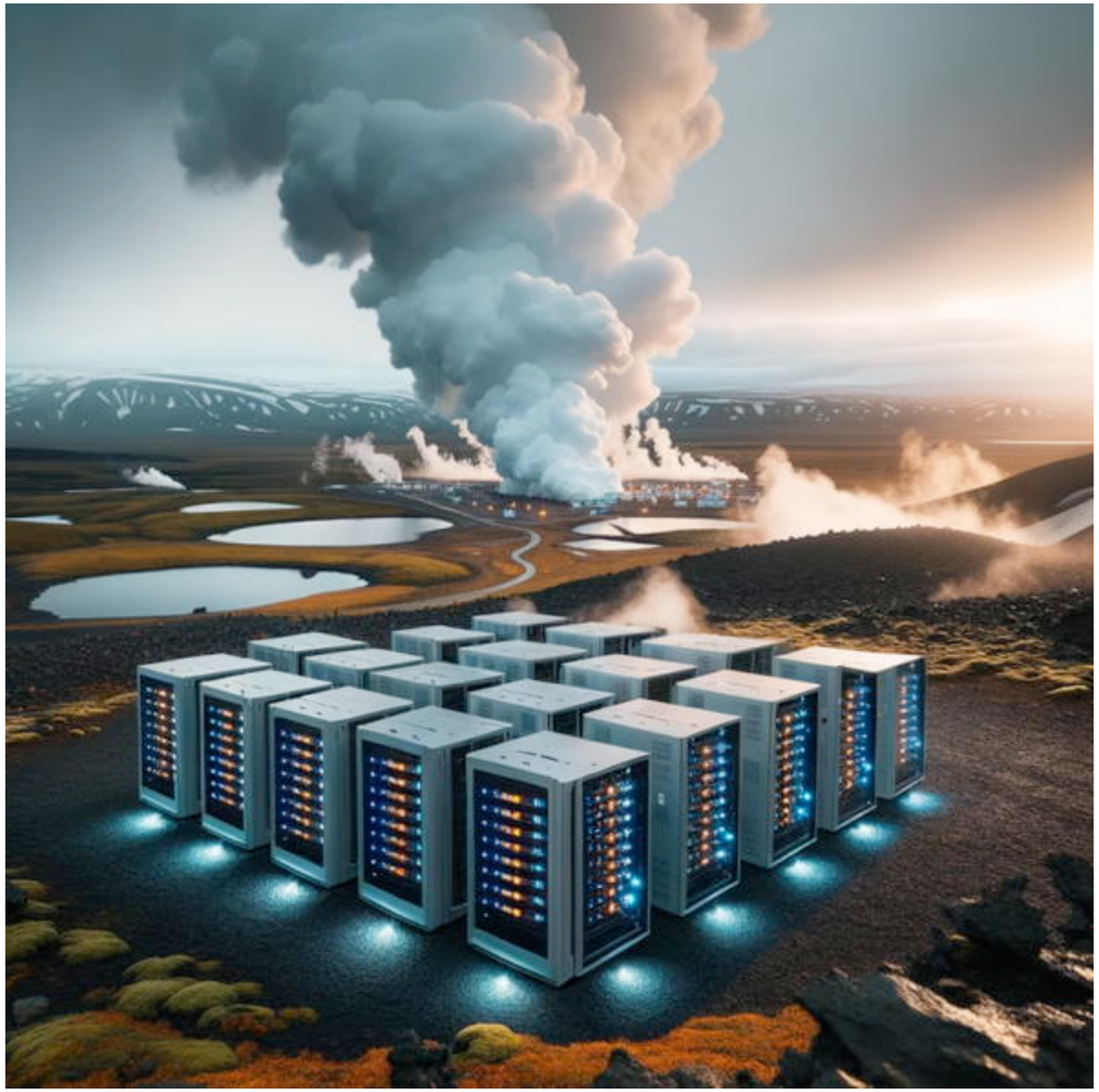
ノイマンアーキテクチャの呪縛との戦い

ノイマンアーキテクチャの呪縛との戦い



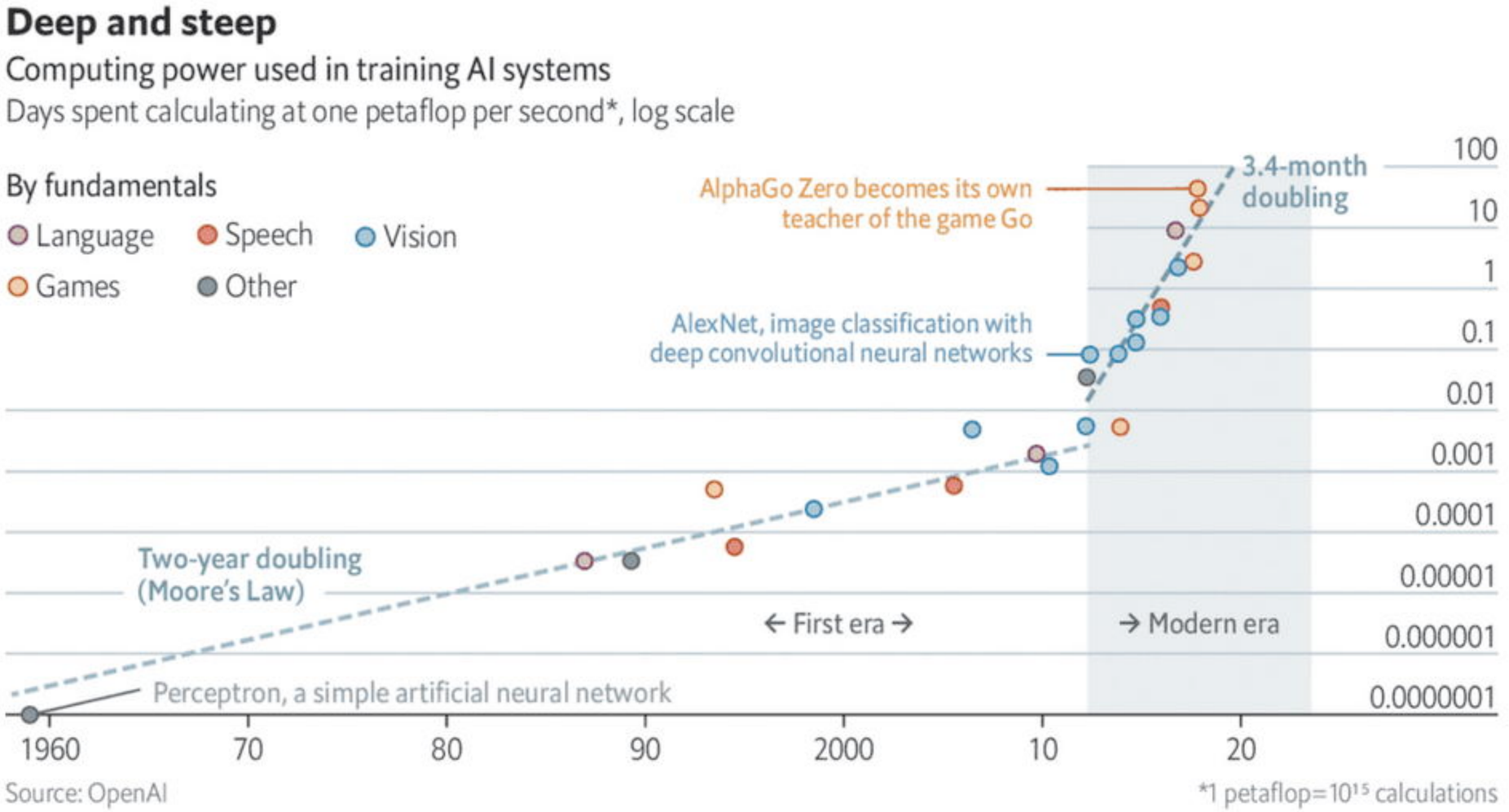
ノイマンアーキテクチャの呪縛との戦い

# ノイマンアーキテクチャの呪縛との戦い



ノイマンアーキテクチャの呪縛との戦い

ノイマンアーキテクチャの呪縛との戦い



本報告の事例はかなり古い
1980年代後半～1990年代前半
のものです。ただし現代の
「AI energy crisis」にも通じる
重要なアプローチです

本日の話題(予稿より)

(1)データフロー・プロセッサ

(2)Cycle-Steel手法

(3)Propellerプロセッサ

時間の関係で、最も重要な(2)から始めます

安心してください

おさらい(釈迦に説法)から始めます

# 参考文献(QRコード撮影準備)

# 参考文献(1)

(絶版になった(;_;)ので出版社編集長の許諾下でPDFをテキストとして使用)

## マイコン技術者スキルアップ事典（→文献1）

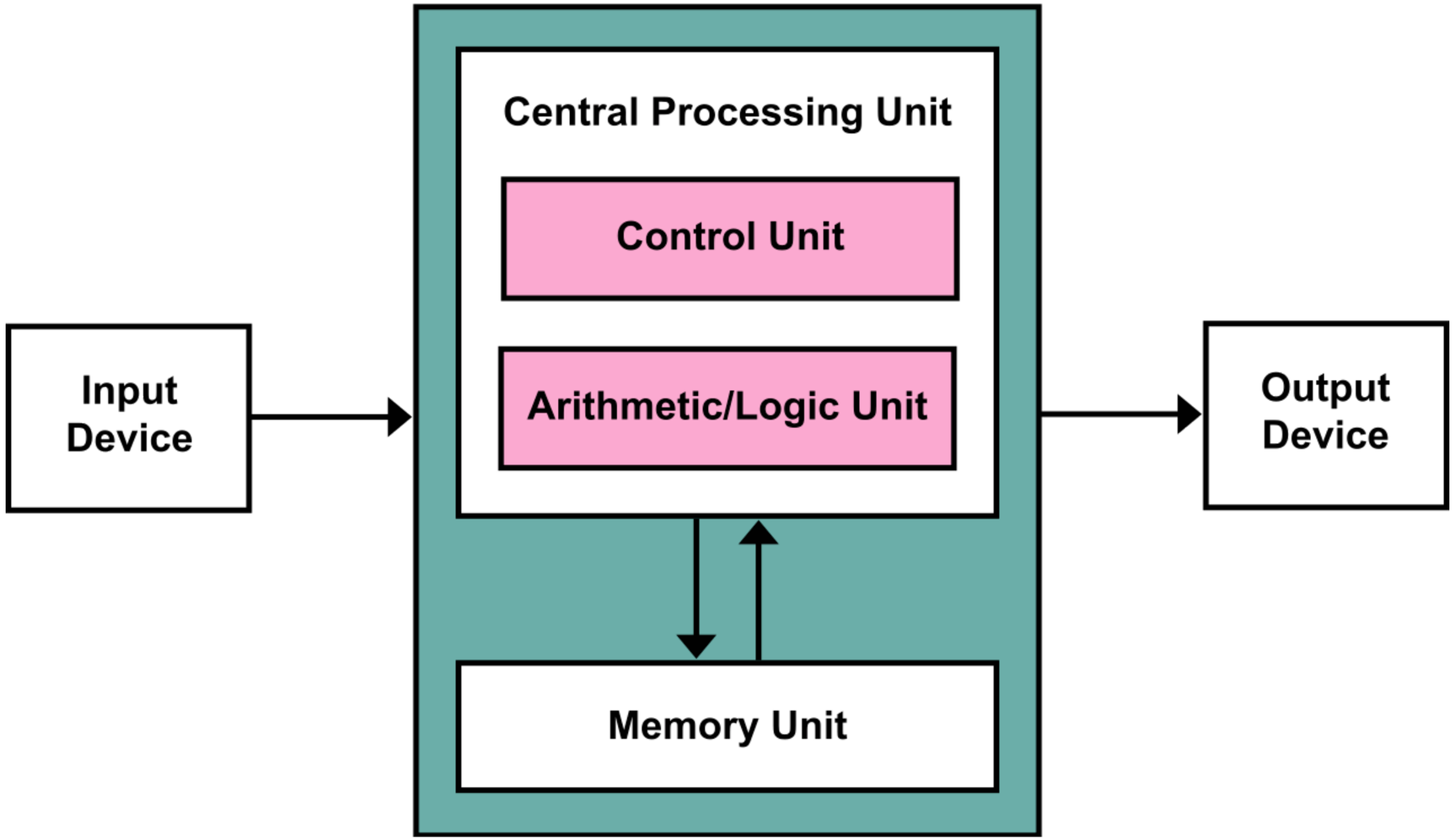
# 参考文献(2)

(絶版になった(;_;)ので出版社編集長の許諾下でPDFをテキストとして使用)

## コンピュータサウンドの世界（→文献2）

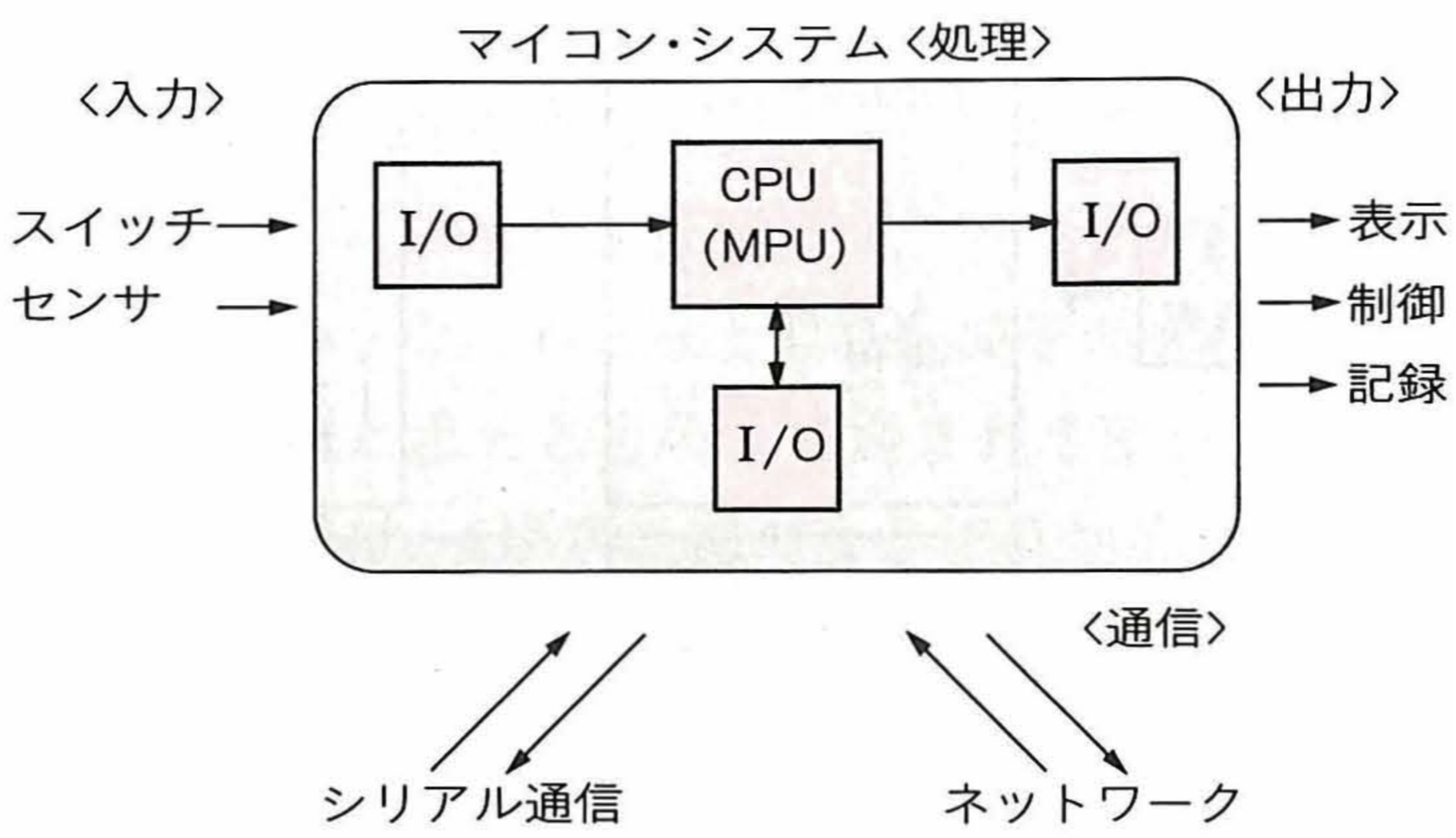
# おさらい(釈迦に説法)

(Wikipedia)



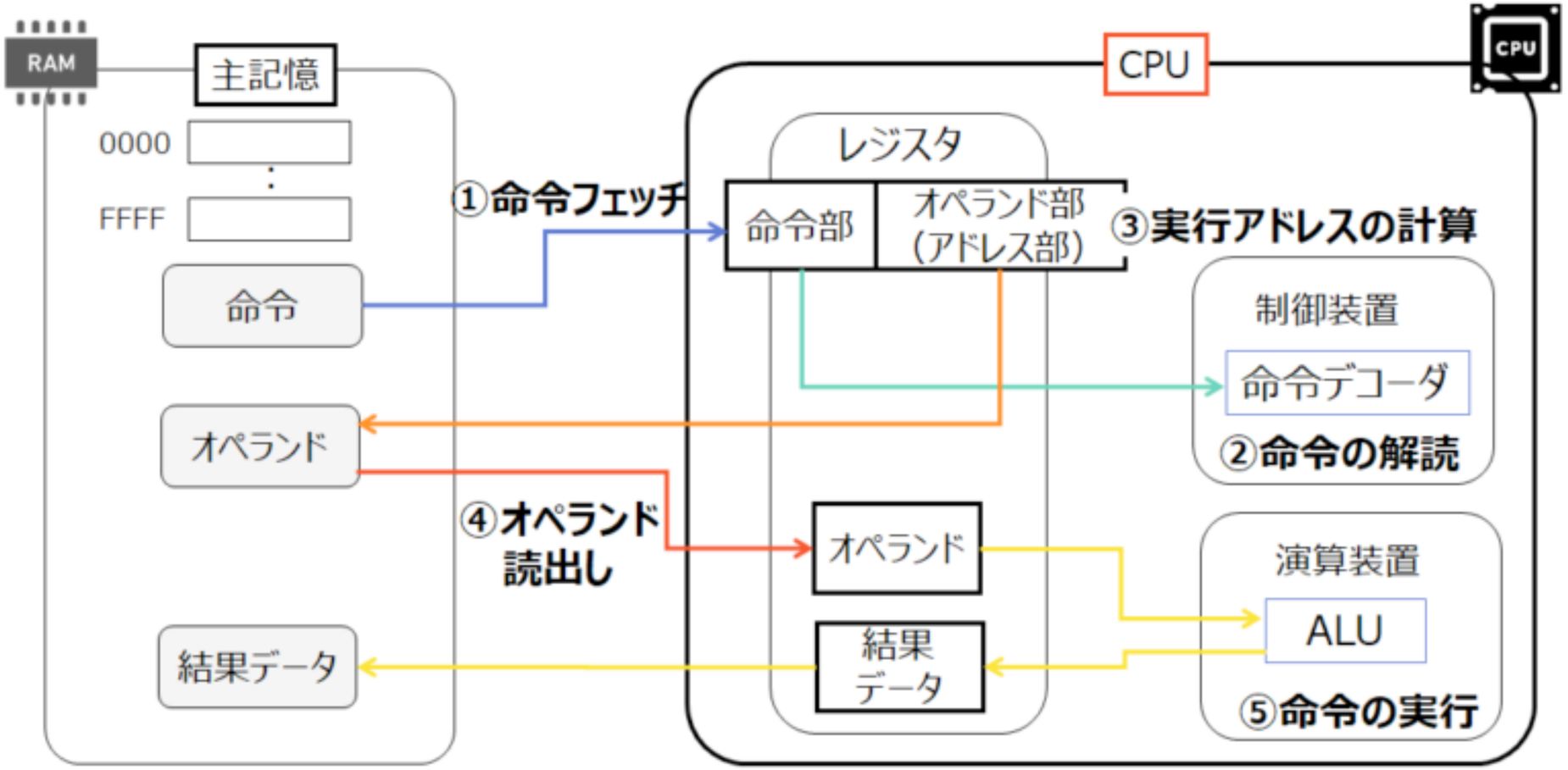

# ノイマン型コンピュータ・アーキテクチャ

（文献1）



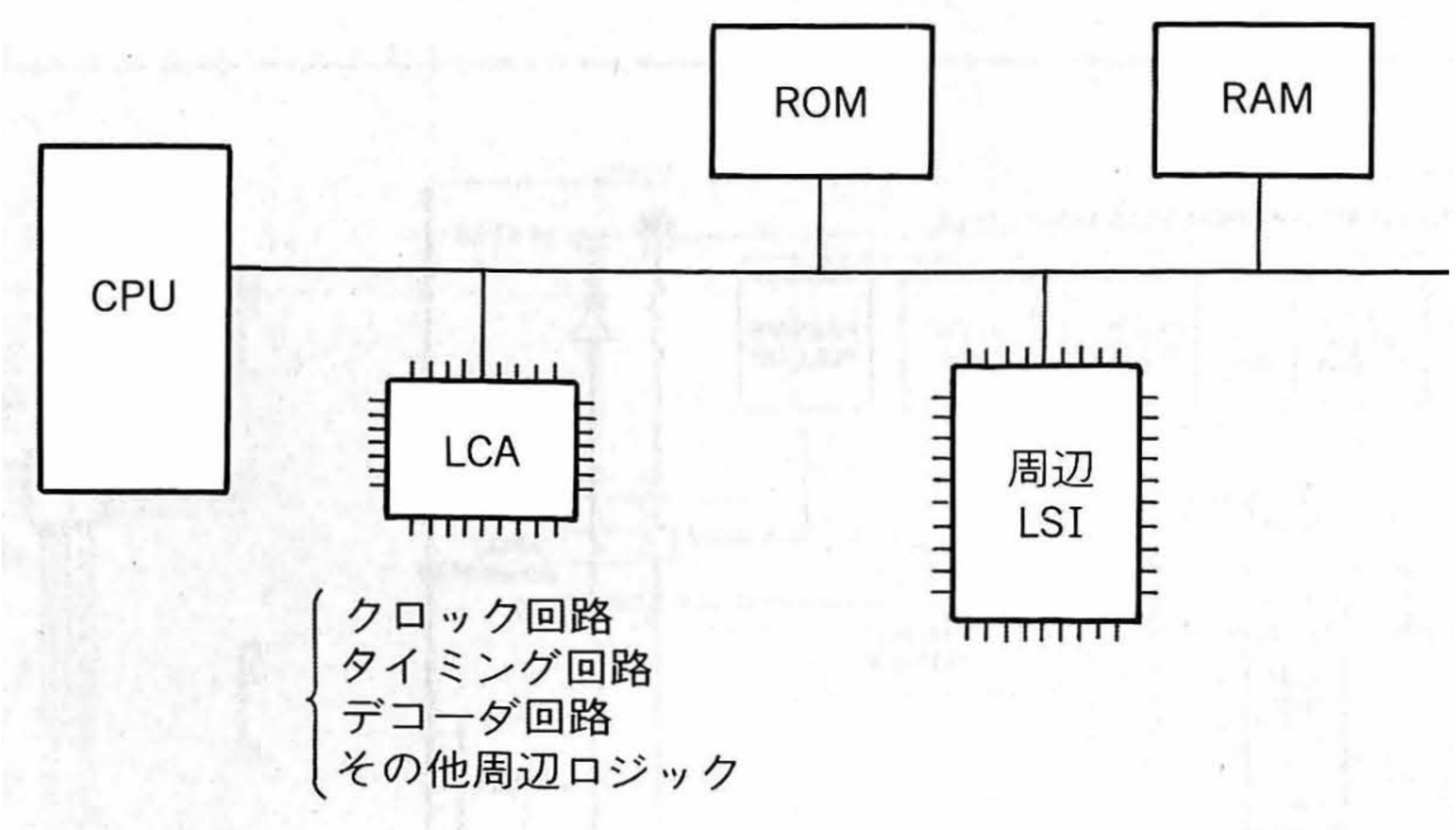

マイコンシステムと外界とのやりとり

ノイマンアーキテクチャの呪縛との戦い



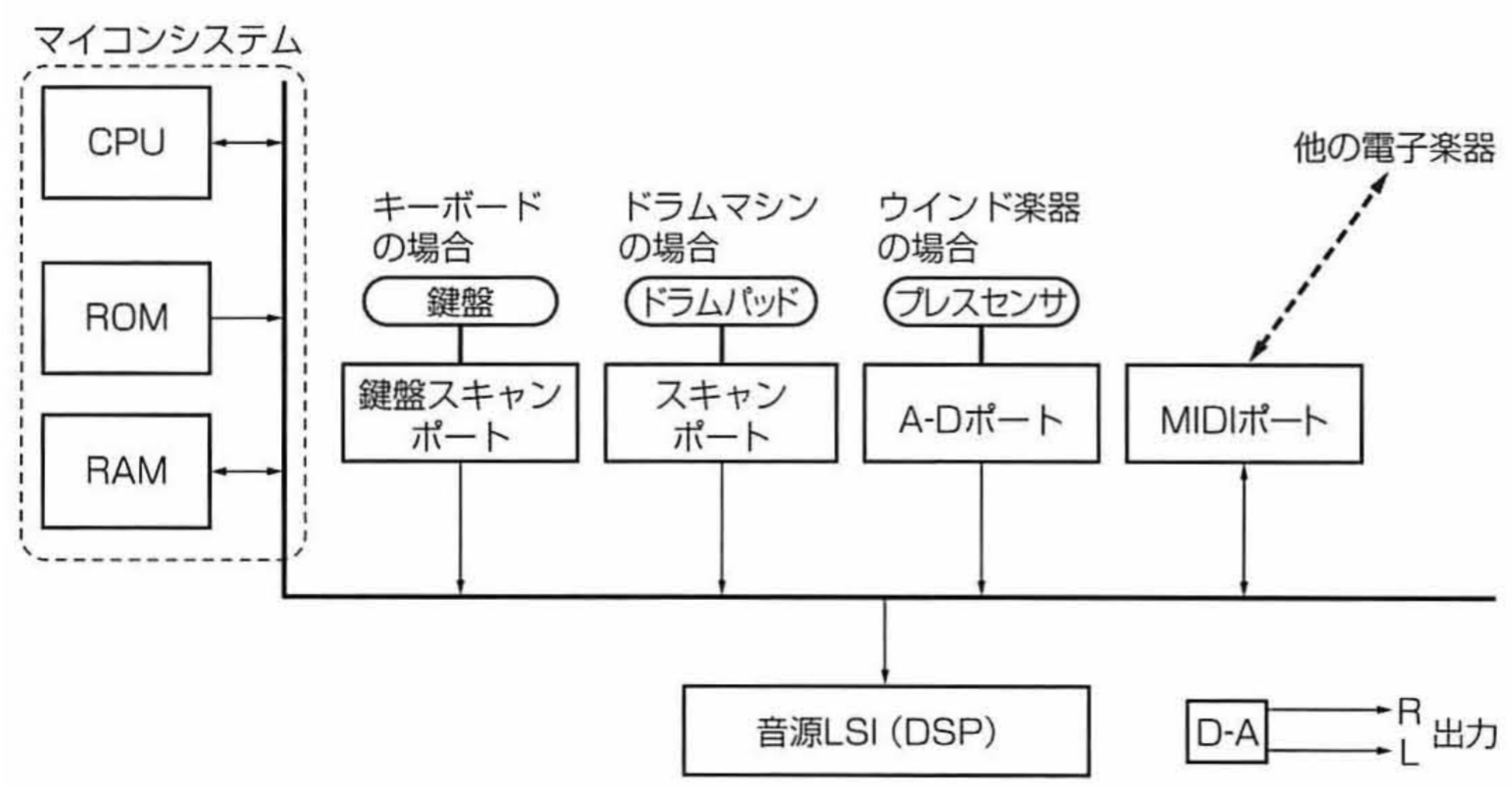

# CPUサイクルはメモリアクセスの嵐

（文献1）



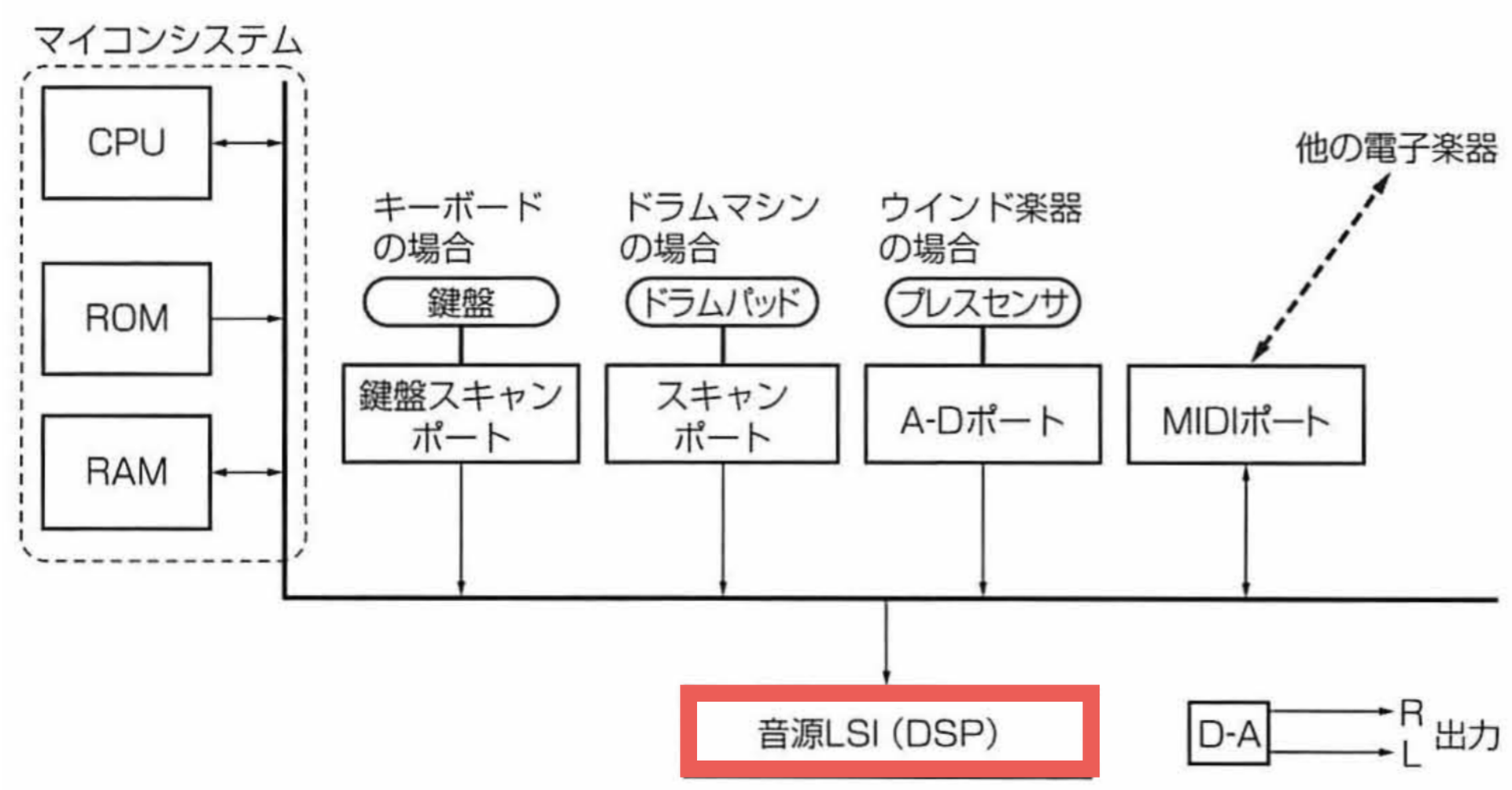

# マイコンシステムの基本構造

（文献2）

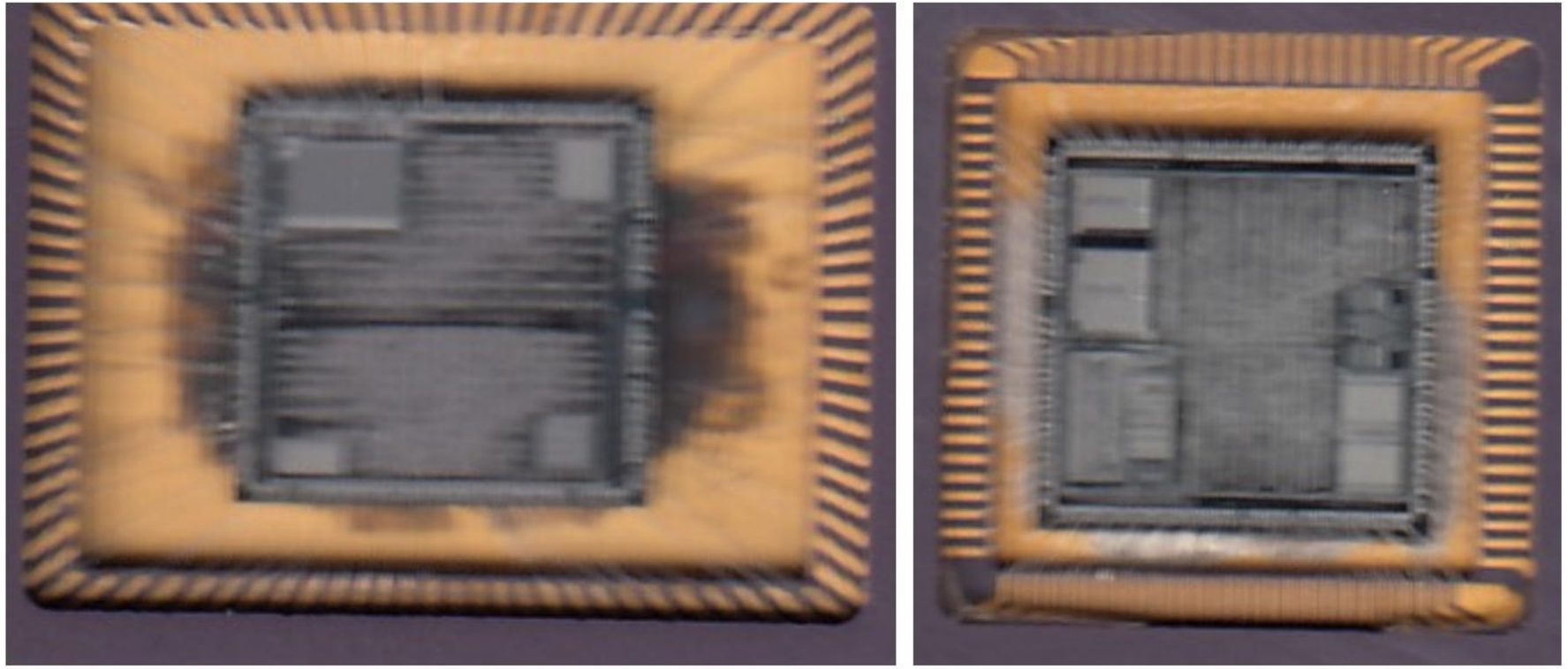



# マイコン応用システム(電子楽器)

（文献2）

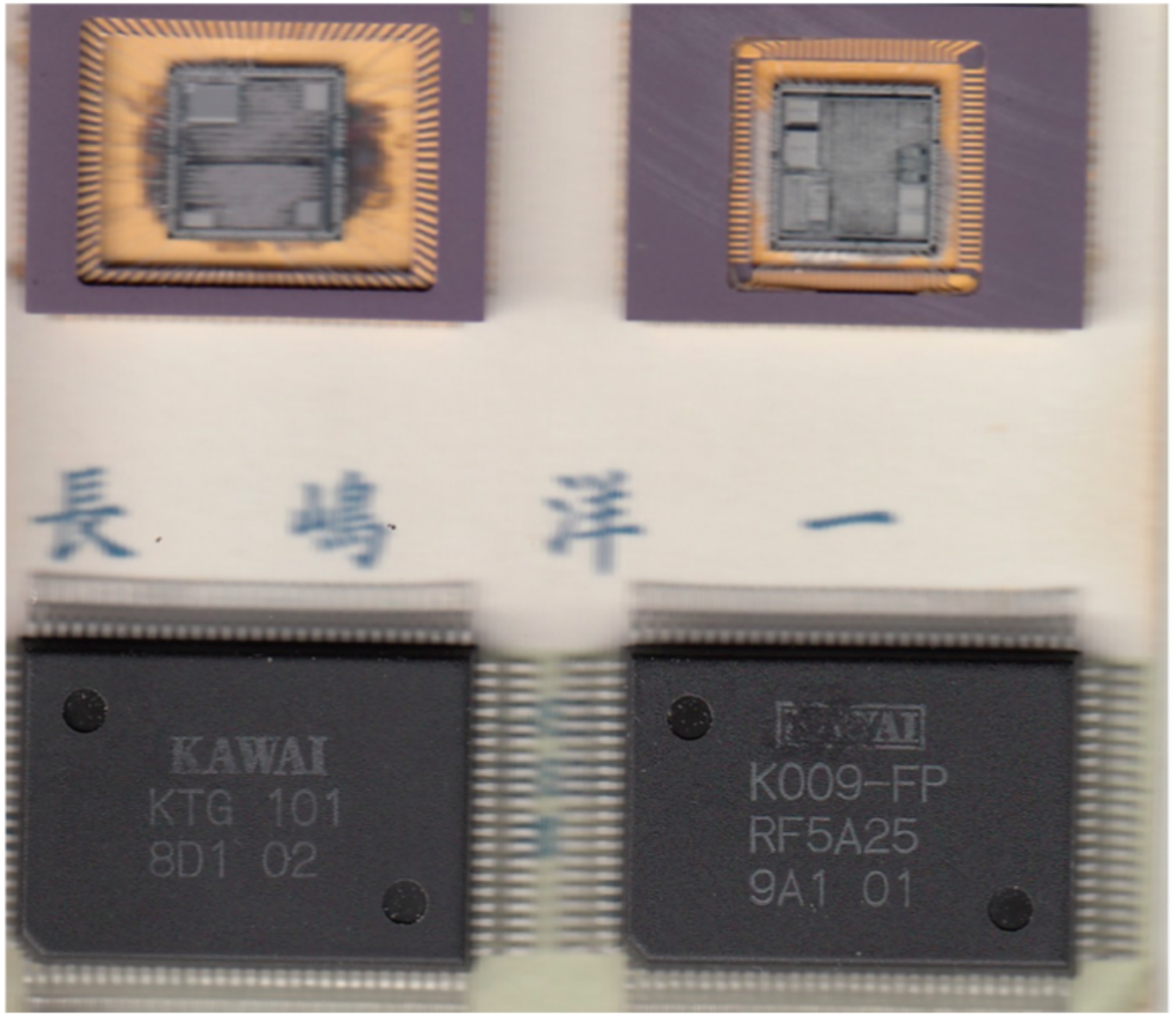



# マイコン応用システム(電子楽器)



# 「音源LSI」の実例(テストサンプル)

RICOH
1.5μm



# （ここだけの話）

ノイマンアーキテクチャの呪縛との戦い

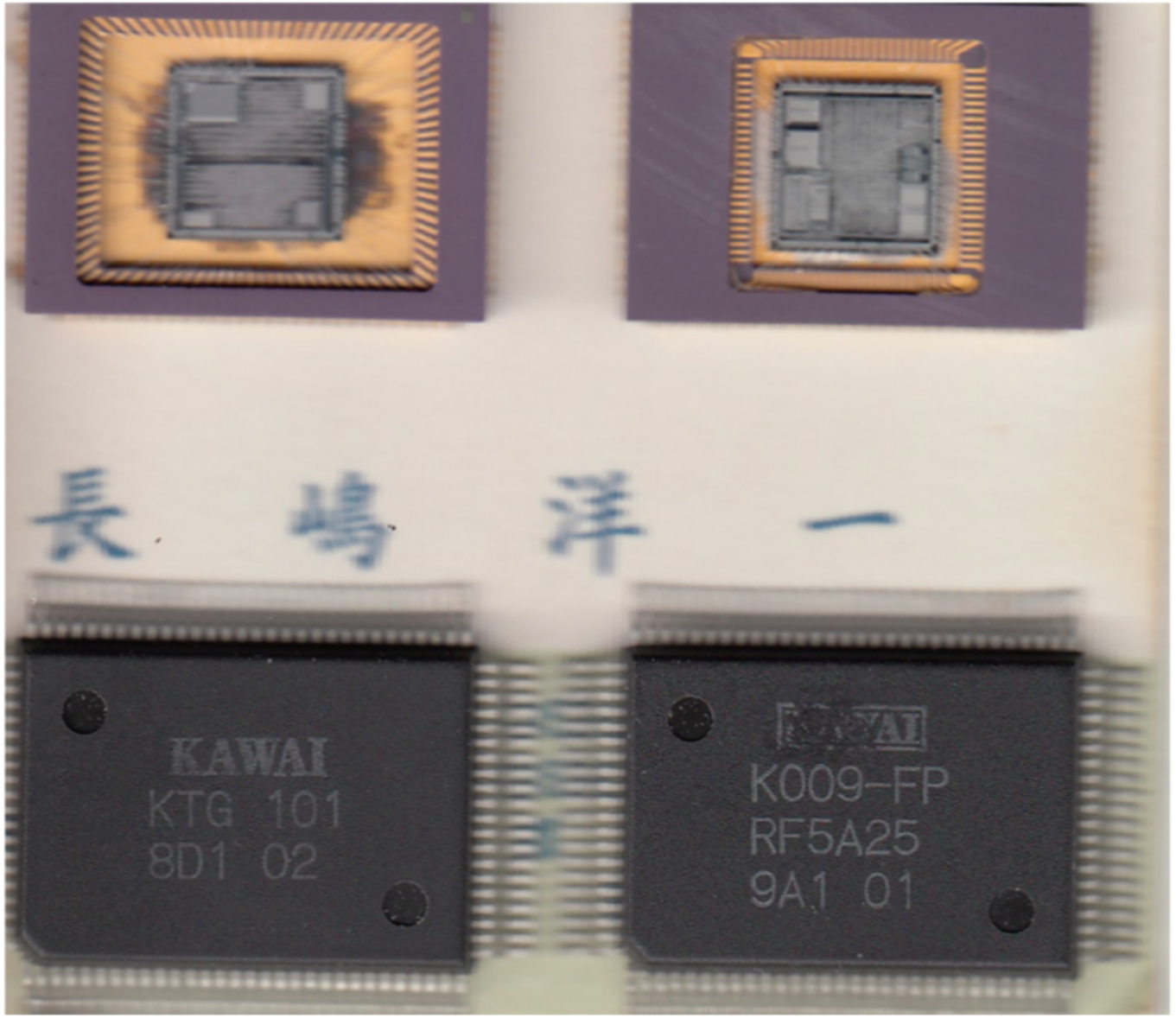
KAWAI 1991

Sound Palette

# 基板上には東芝製CPUがある

KAWAI 1991

Sound Palette




RICOH

1.2μm

ノイマンアーキテクチャの呪縛との戦い

KAWAI

MS710


1992

# 基板上にCPUが無い!!
# →業界をザワつかせる

KAWAI
MS710


1992

# いよいよ本論

# 工学的に言えば・・・

システムの性能向上のために

→ システムクロックをより高速化

→ メモリアクセスをより短時間化

↓

コストアップの方向(;_;)

# 工学的に言えば・・・

システムの性能向上のために

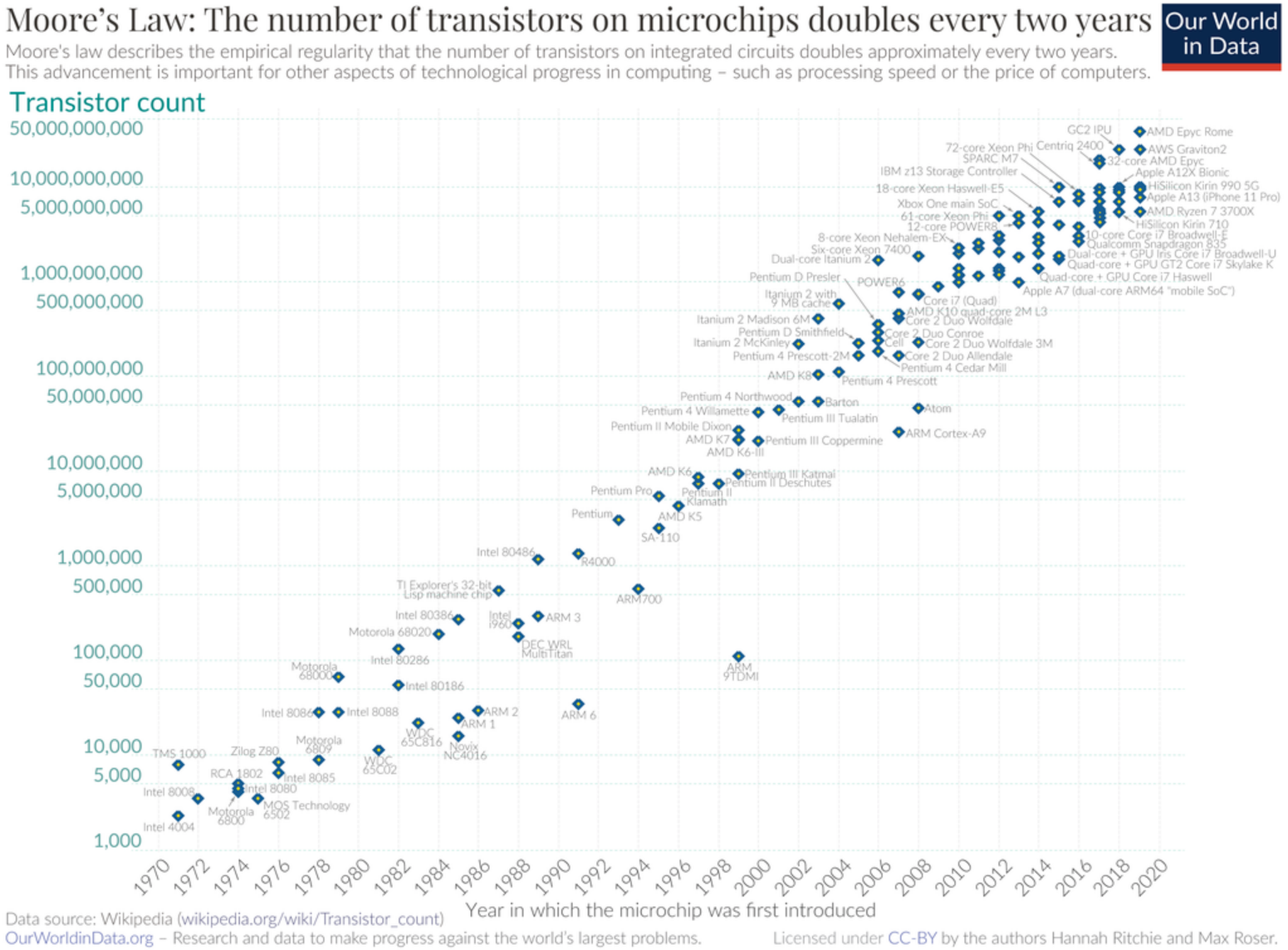
→ ミクロンルールをより微細化

→ 電源/信号をより低電圧化

↓

コストアップの方向(;_;)

ノイマンアーキテクチャの呪縛との戦い




# ムーアの法則

# 時間学的アプローチ(本報告)

システムの性能向上のために

→ システムクロックは同じまま

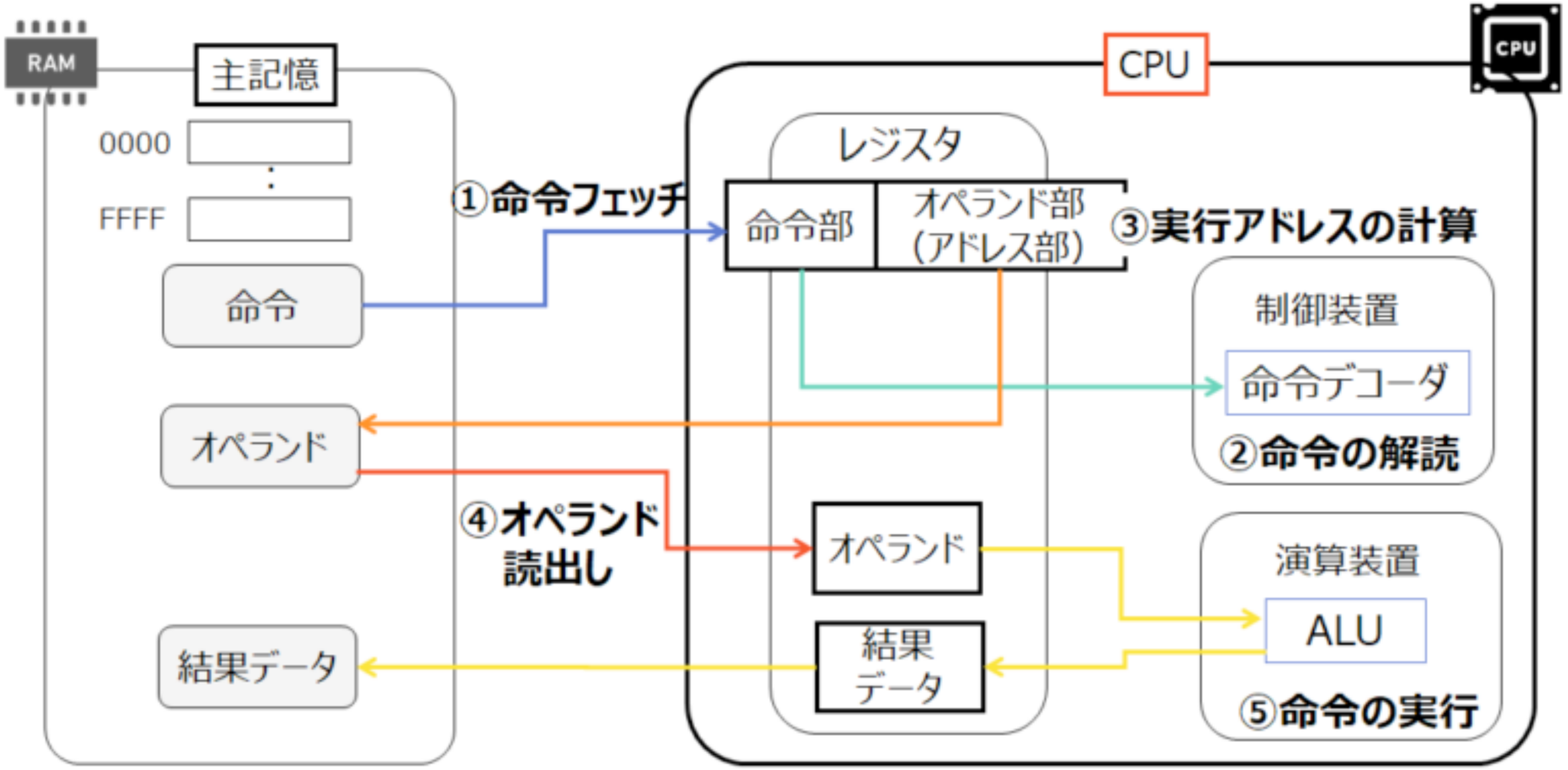
→ メモリアクセスは同じまま

↓

競争力アップ(^o^)

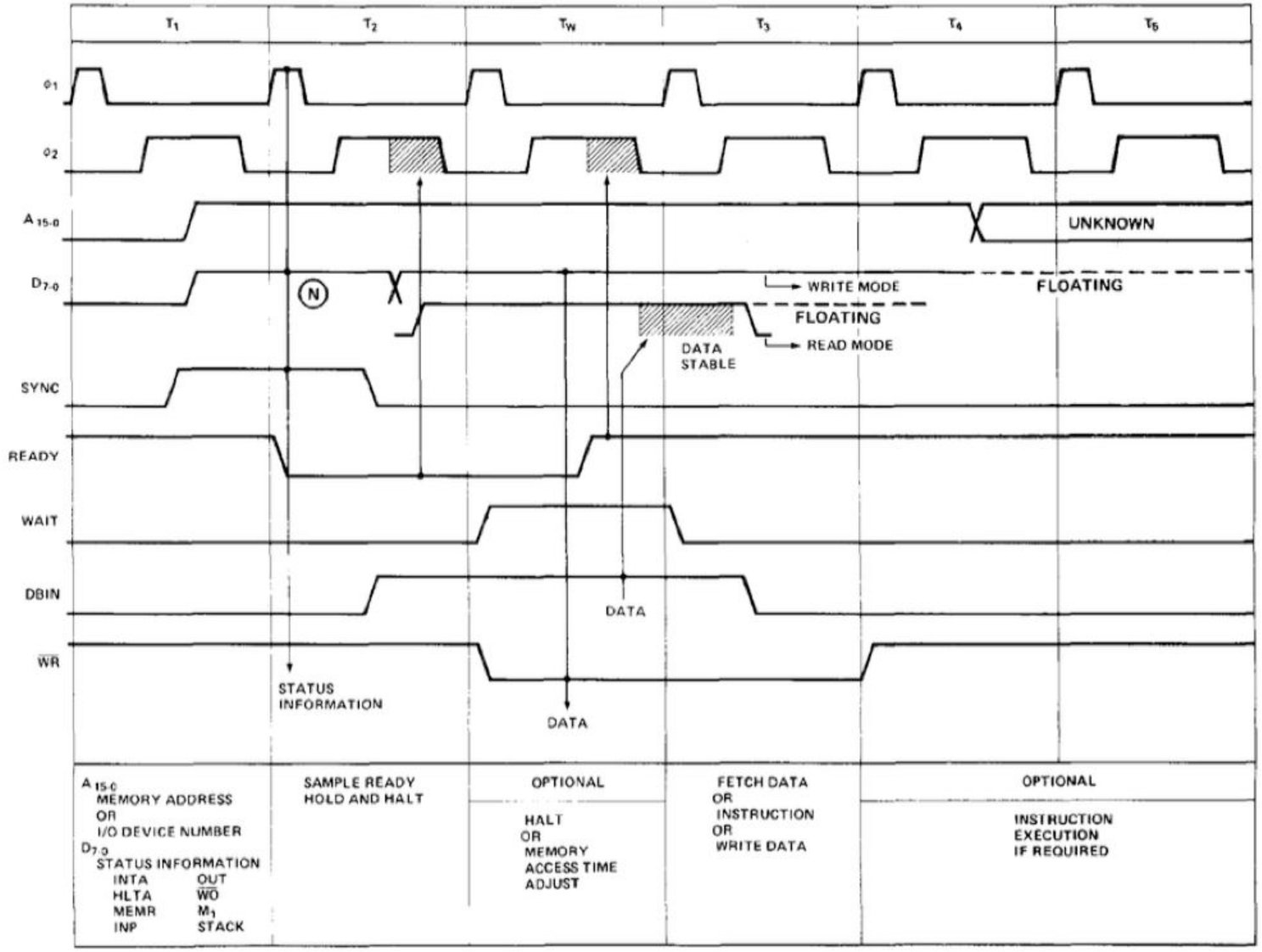
# 1980年代後半～1990年代前半の民生機器CPUとは・・・

- インテル8080
- ザイログZ80
- モトローラ6809
- ロックウェル6502

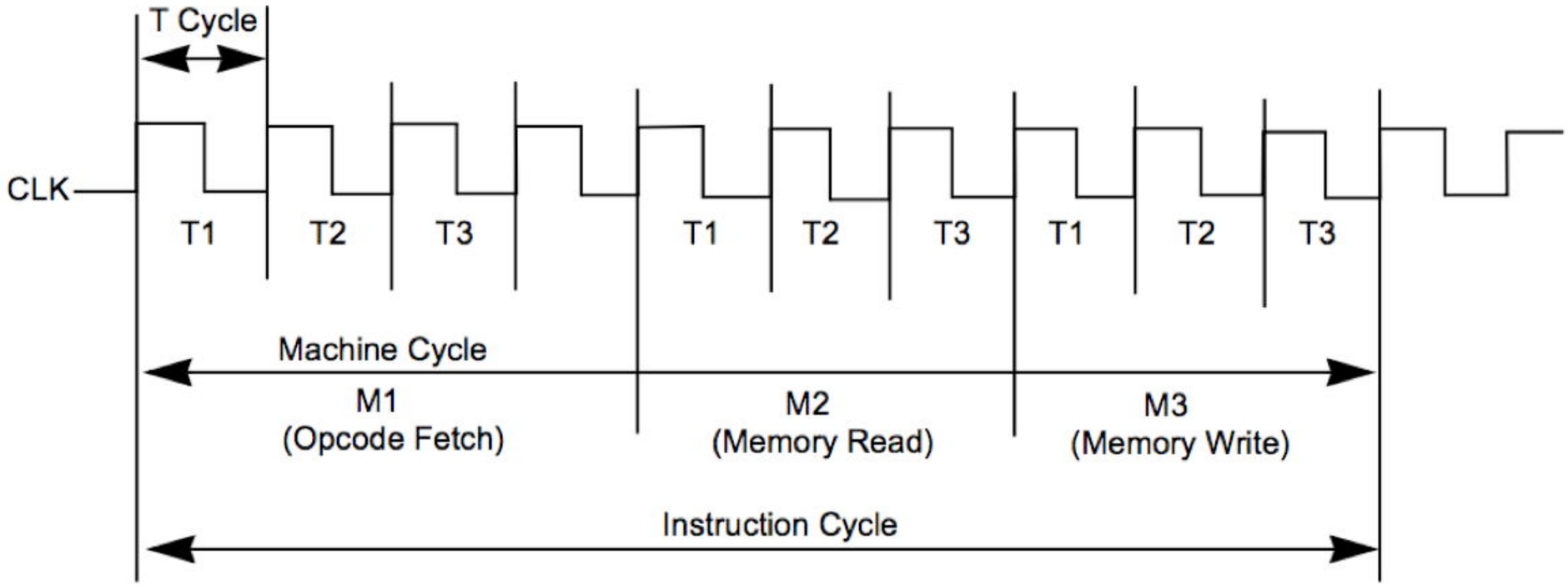

RAM
主記憶
0000
FFFF
命令
オペランド
結果データ
①命令フェッチ
④オペランド
読出し
CPU
CPU
レジスタ
命令部
オペランド部
(アドレス部)
③実行アドレスの計算
制御装置
命令デコーダ
②命令の解読
オペランド
結果
データ
演算装置
ALU
⑤命令の実行

# CPUサイクル(再)

# インテル8080 のCPUサイクル

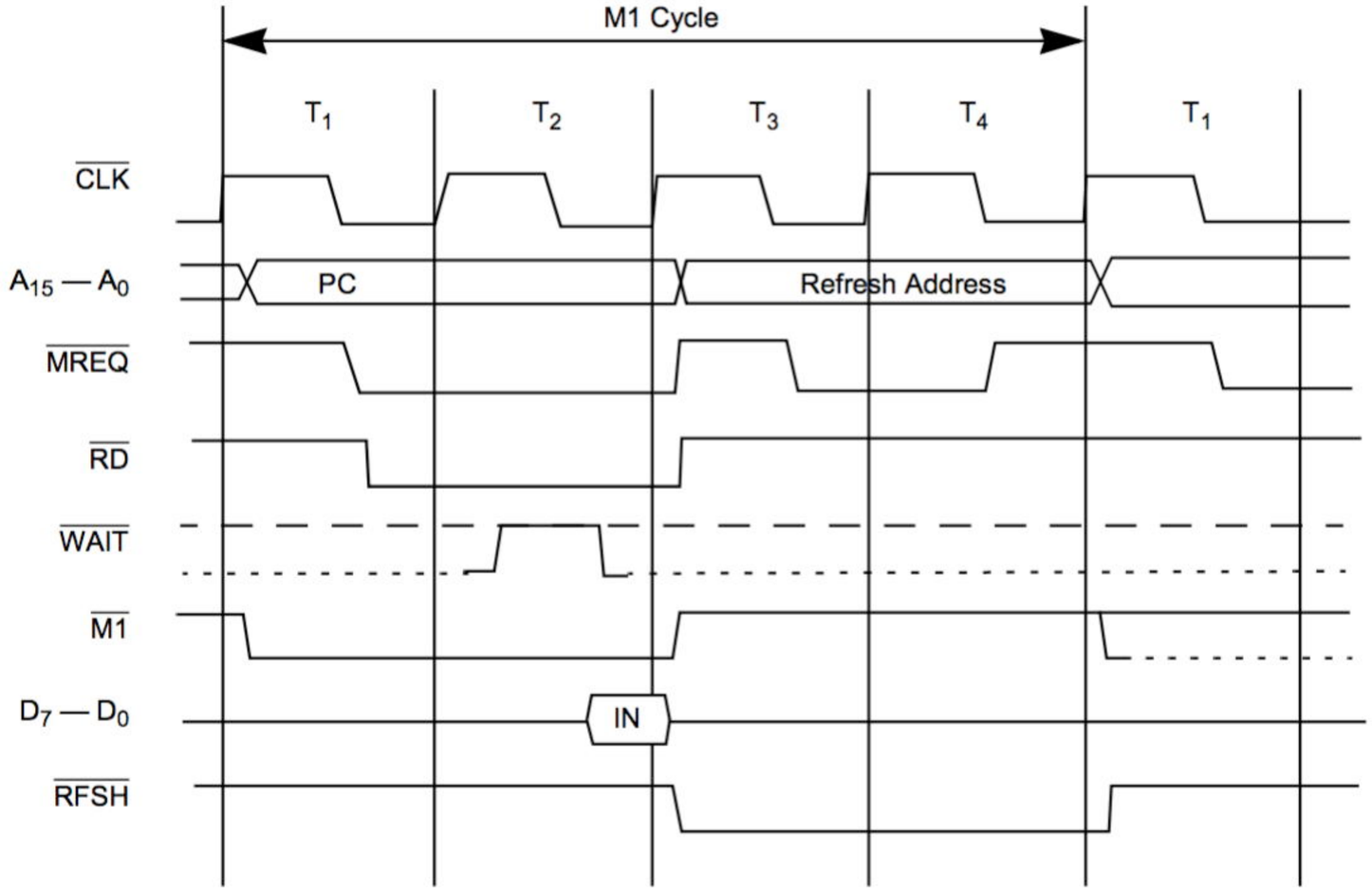


# ザイログZ80 のCPUサイクル



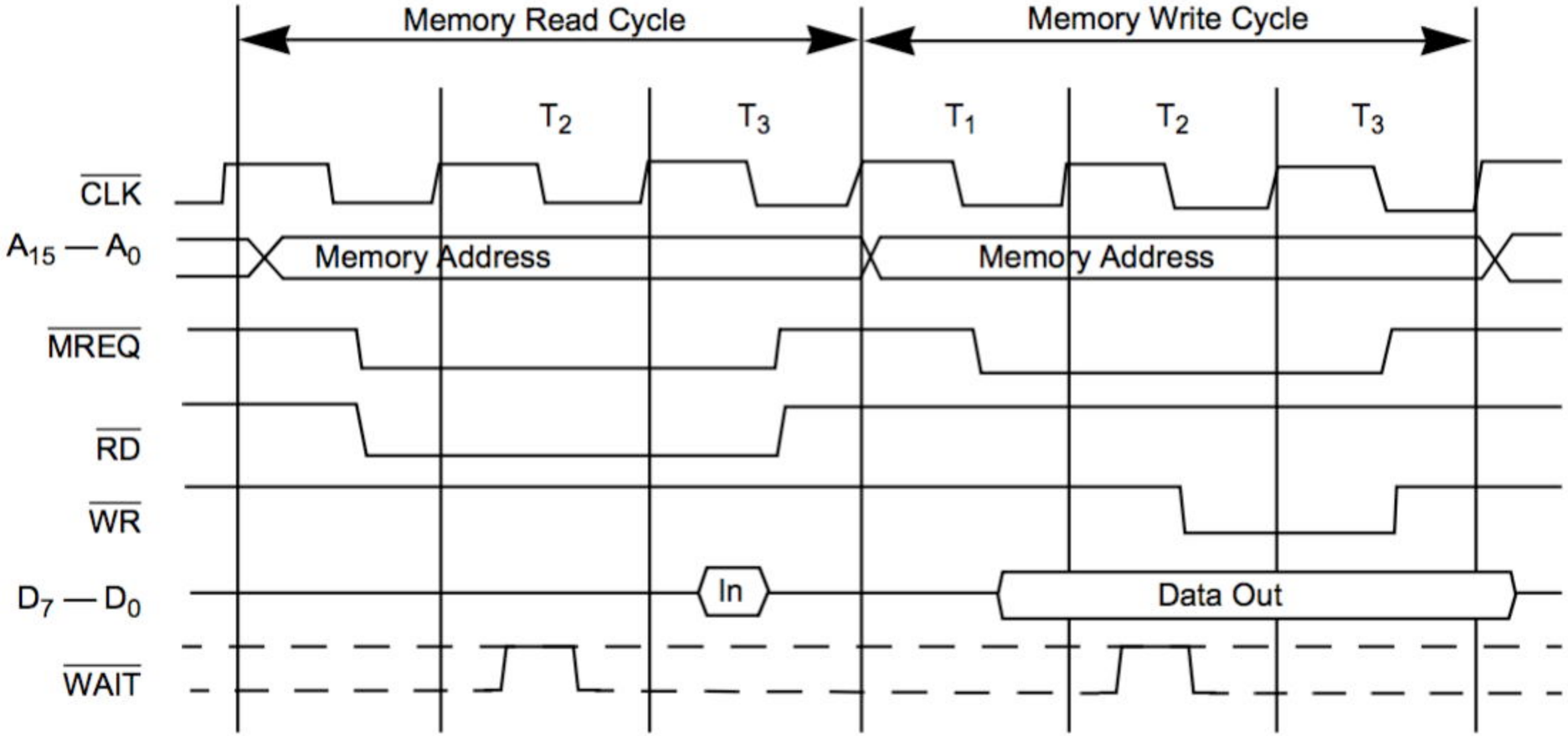

**Basic CPU Timing Example**

# ザイログZ80 のCPUサイクル



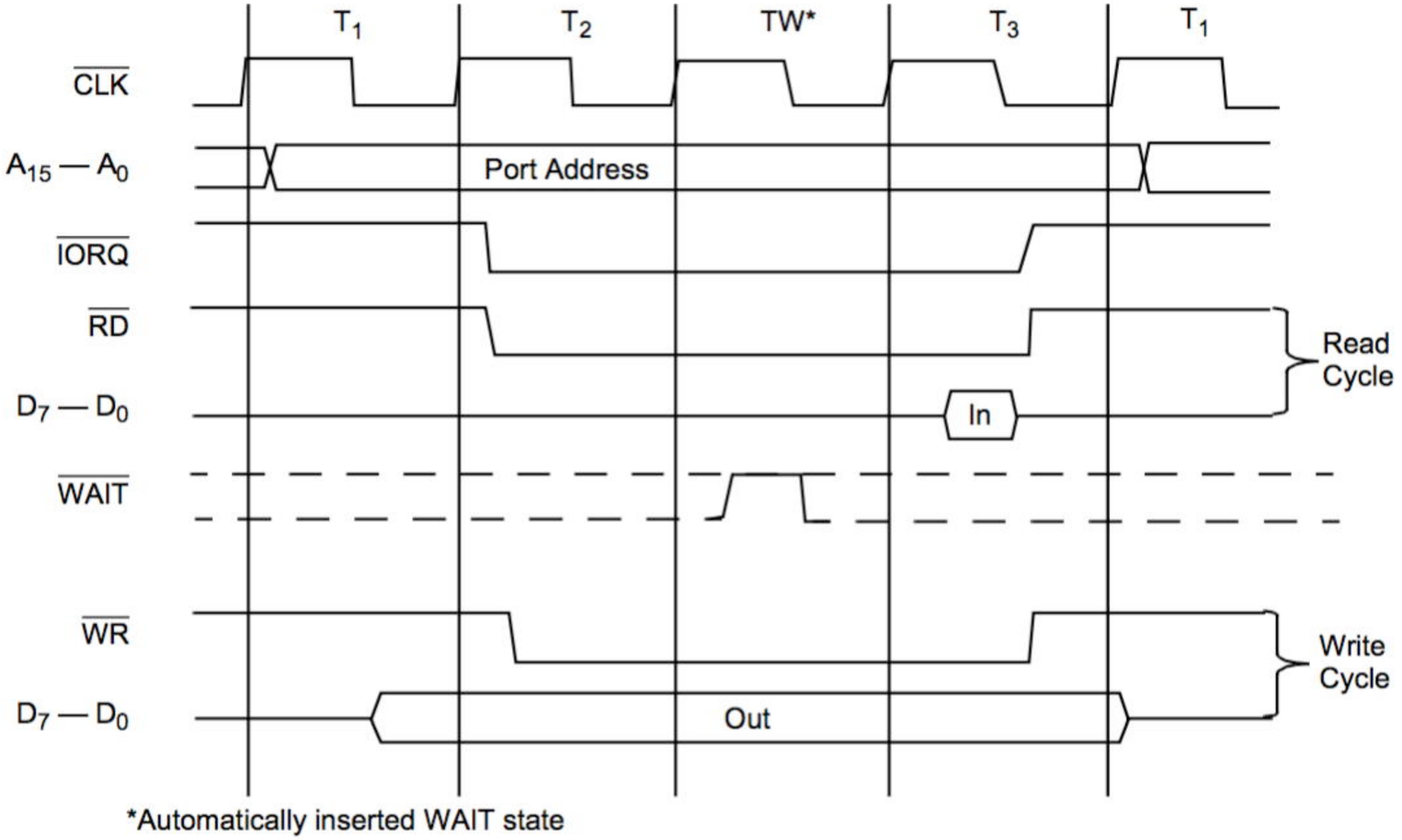

**Instruction Op Code Fetch**

# ザイログZ80 のCPUサイクル



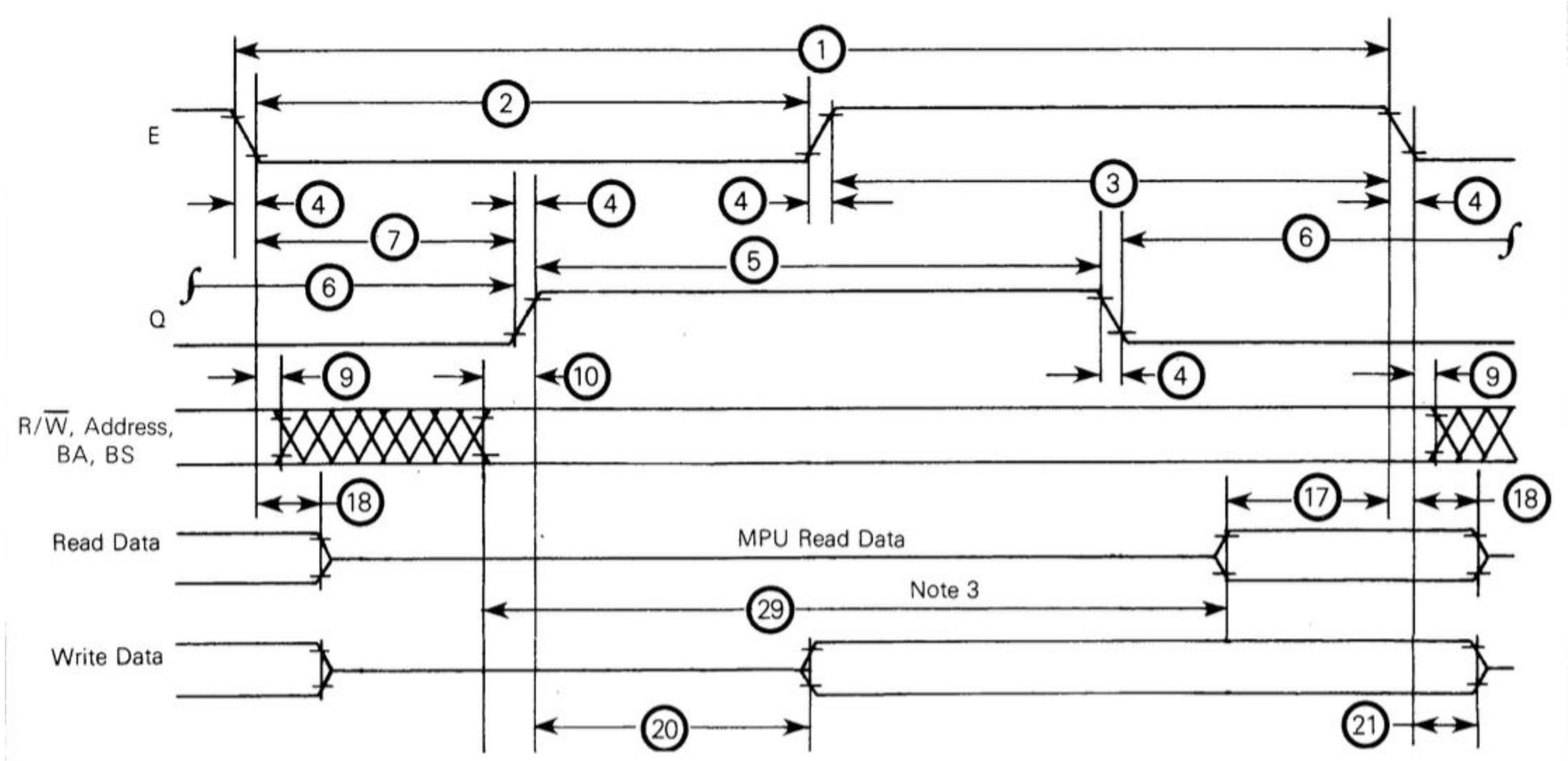

**Memory Read or Write Cycle**

# ザイログZ80 のCPUサイクル

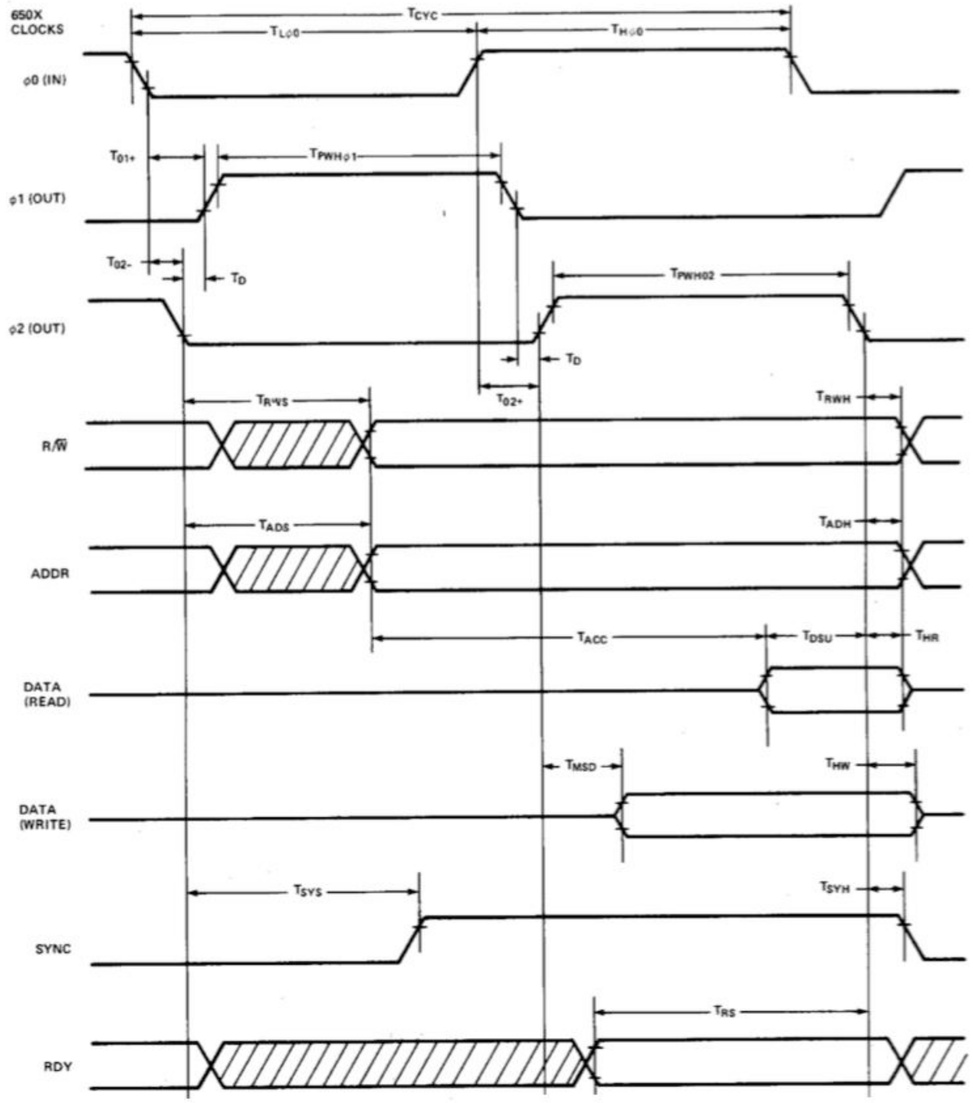



**Input or Output Cycles**

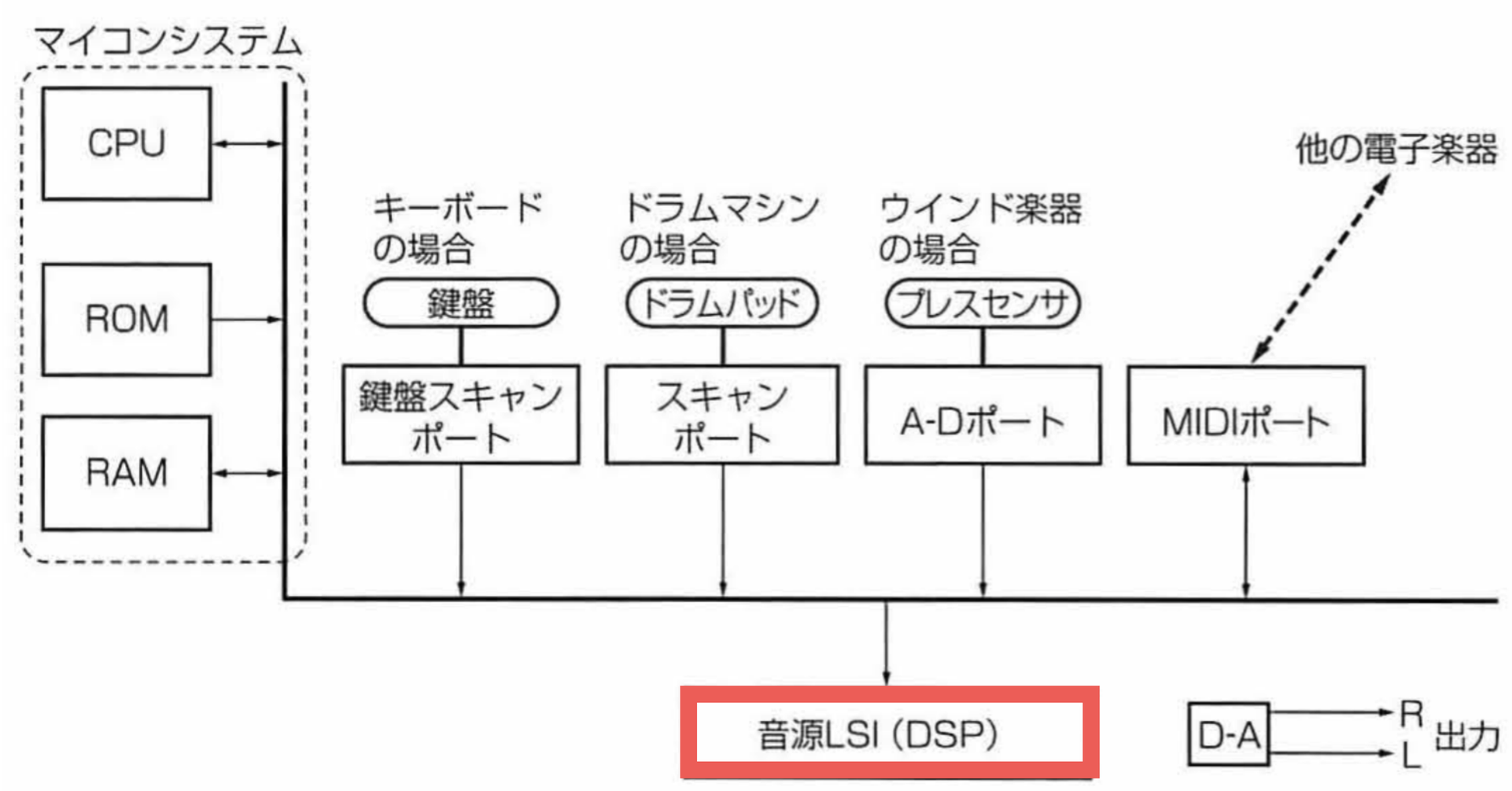
# モトローラ6809 のCPUサイクル



# ロックウェル6502 のCPUサイクル



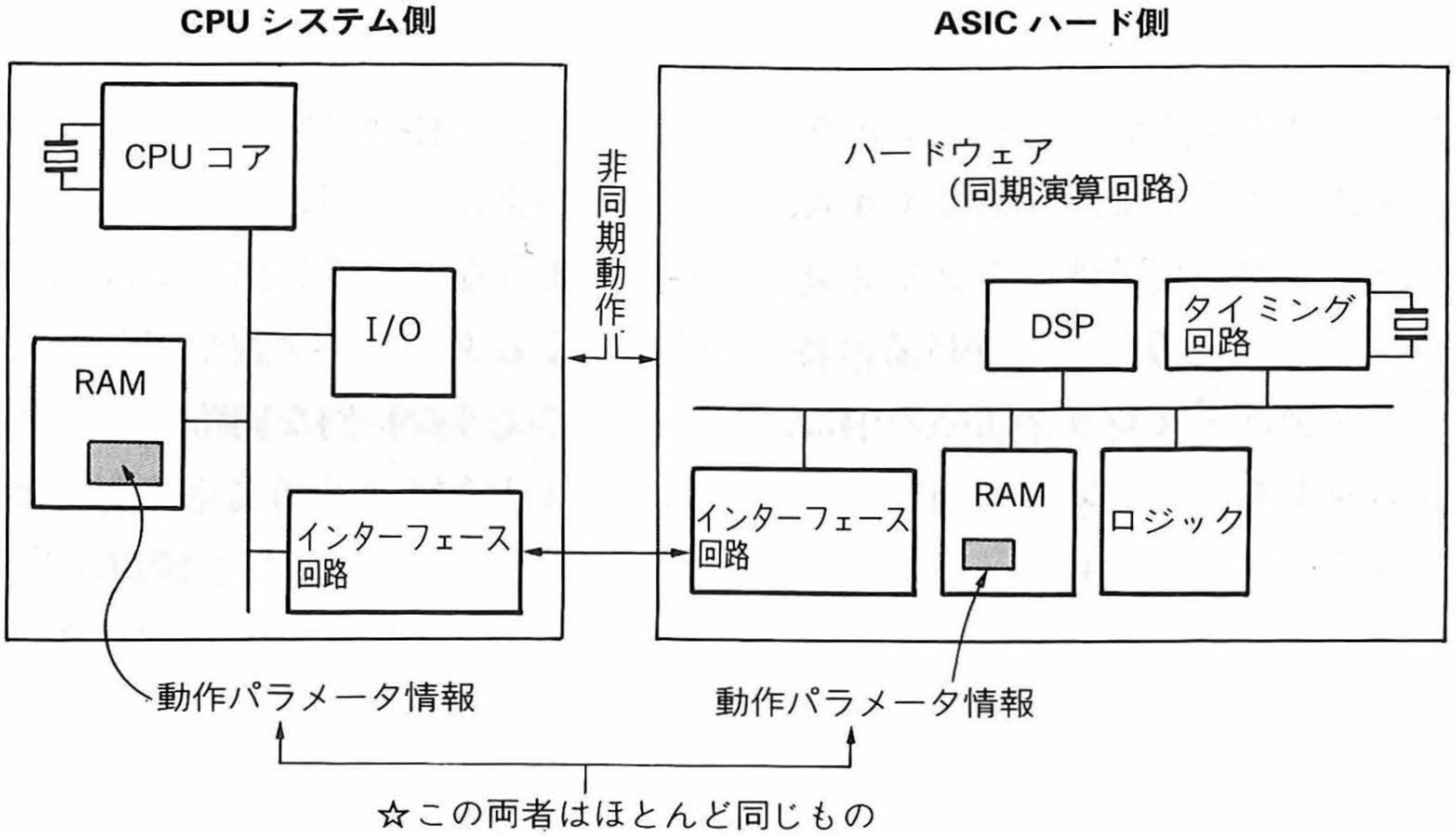
（文献2）




# 電子楽器システム(再)

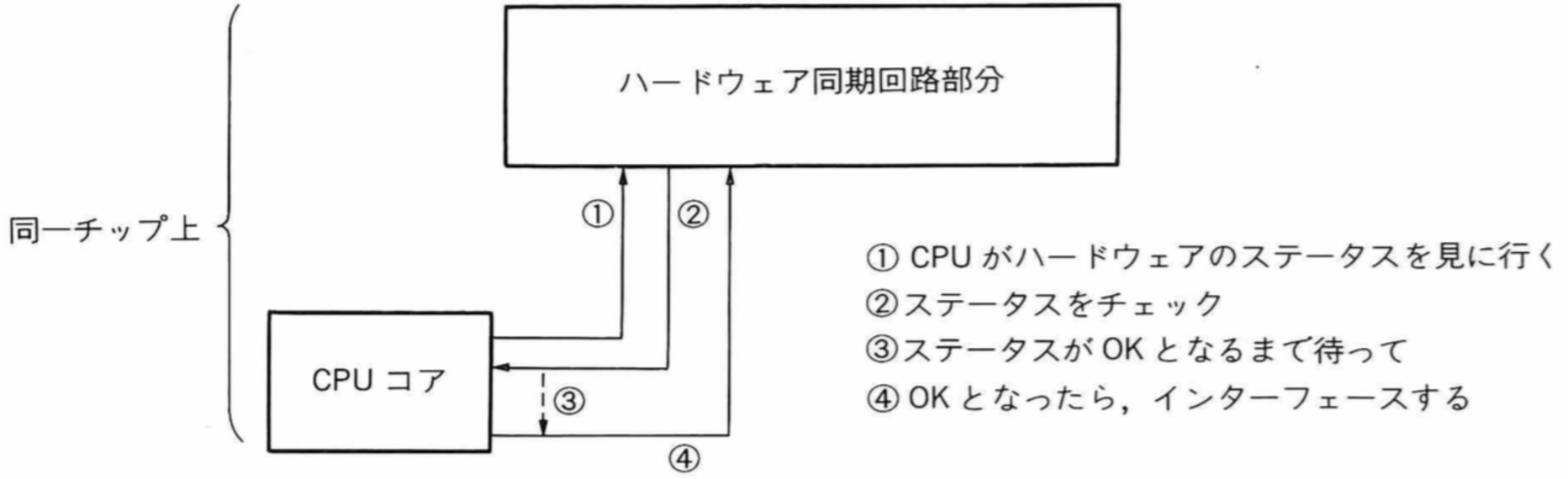
ノイマンアーキテクチャの呪縛との戦い

（文献1）

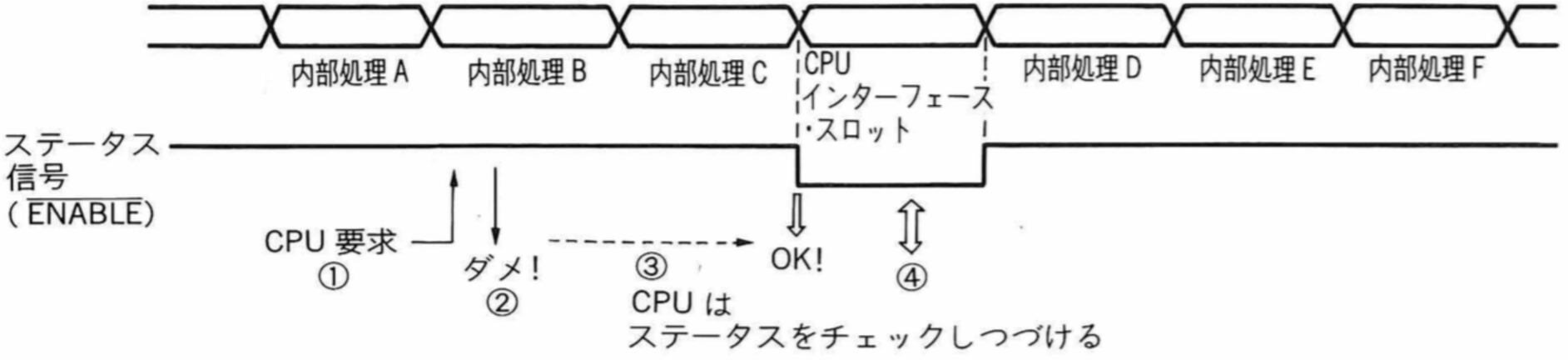


（文献1）

非同期待ち合わせの例（I）

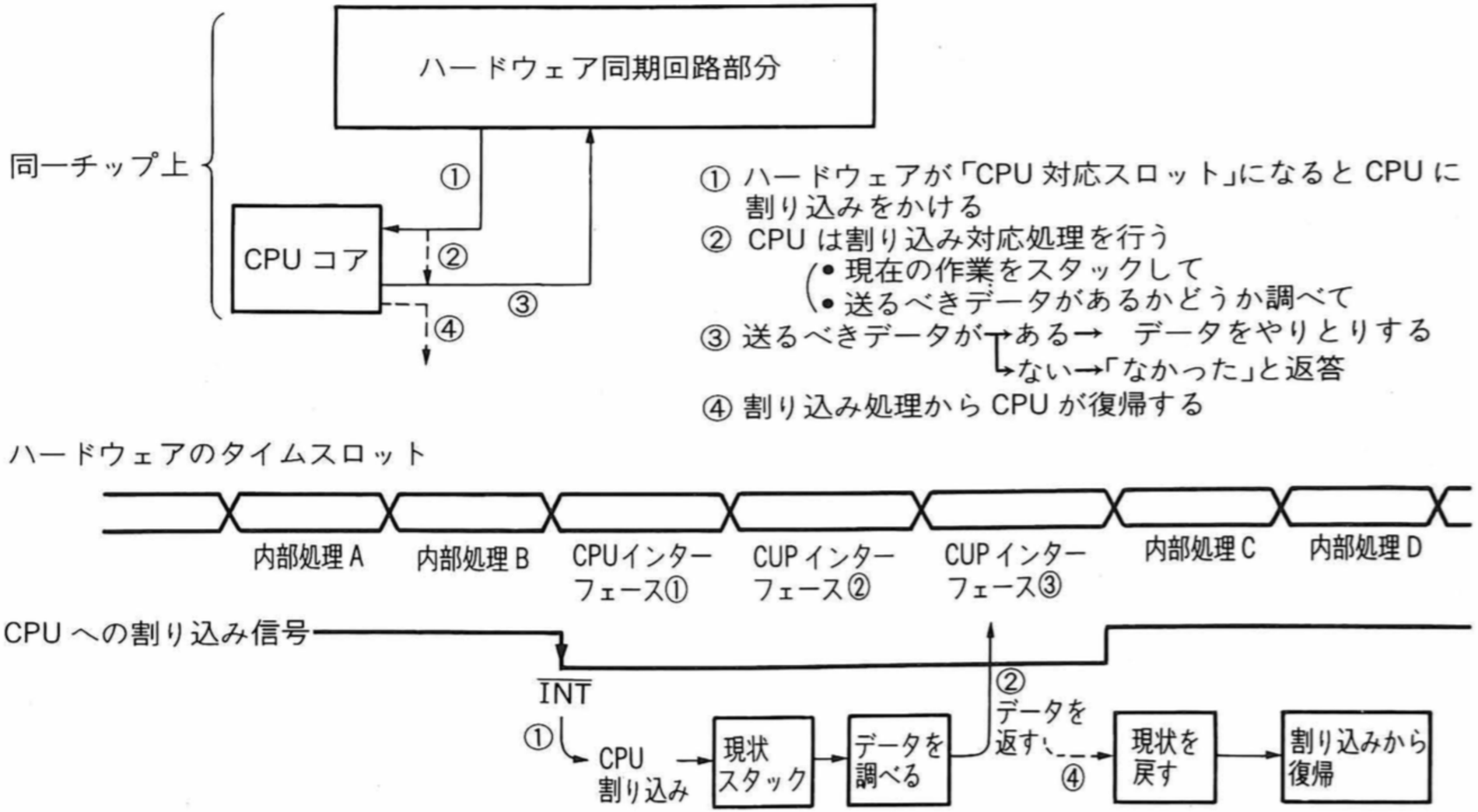



ハードウェアのタイムスロットの状態



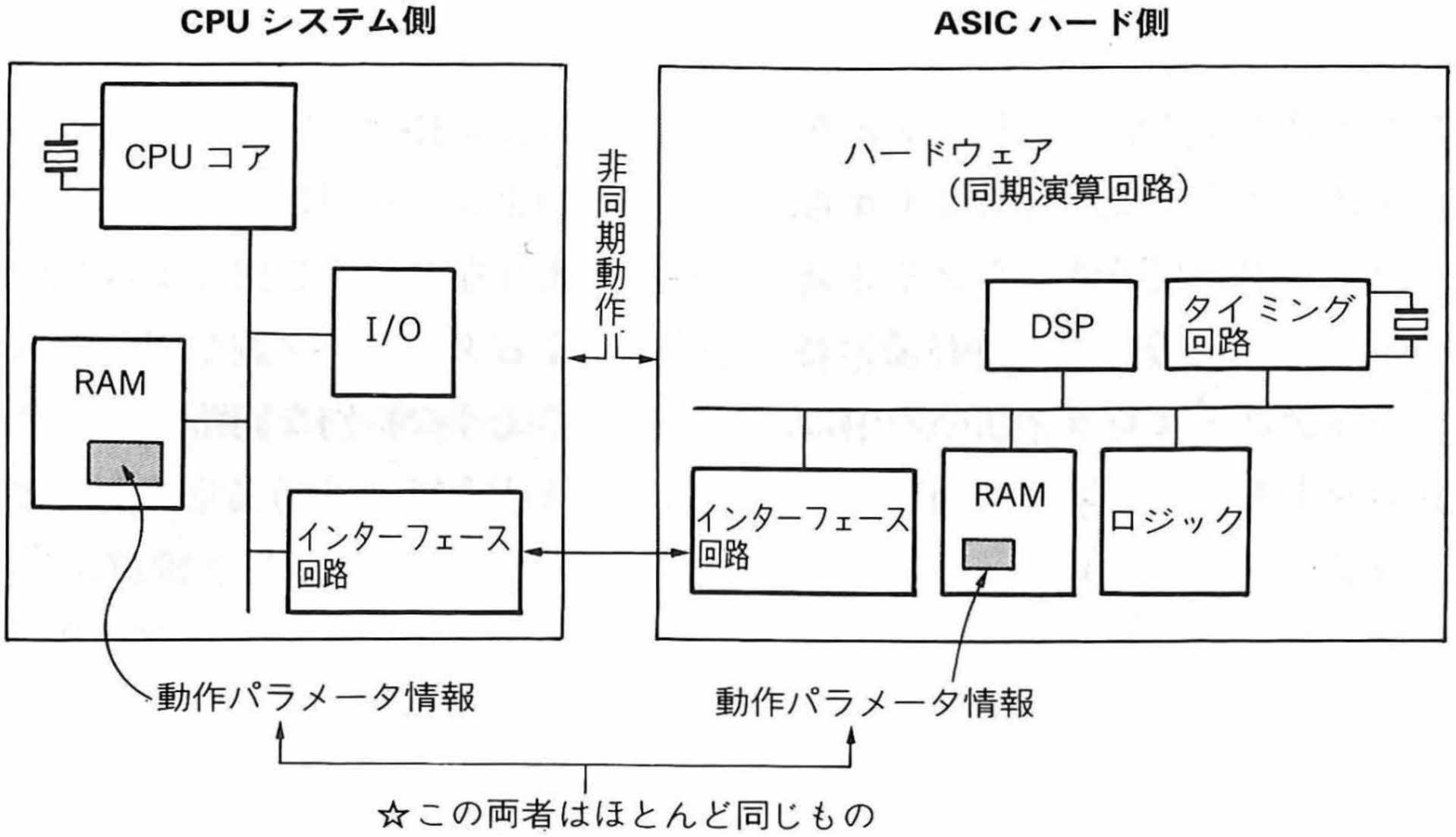
（文献1）

## 非同期待ち合わせの例 (2)

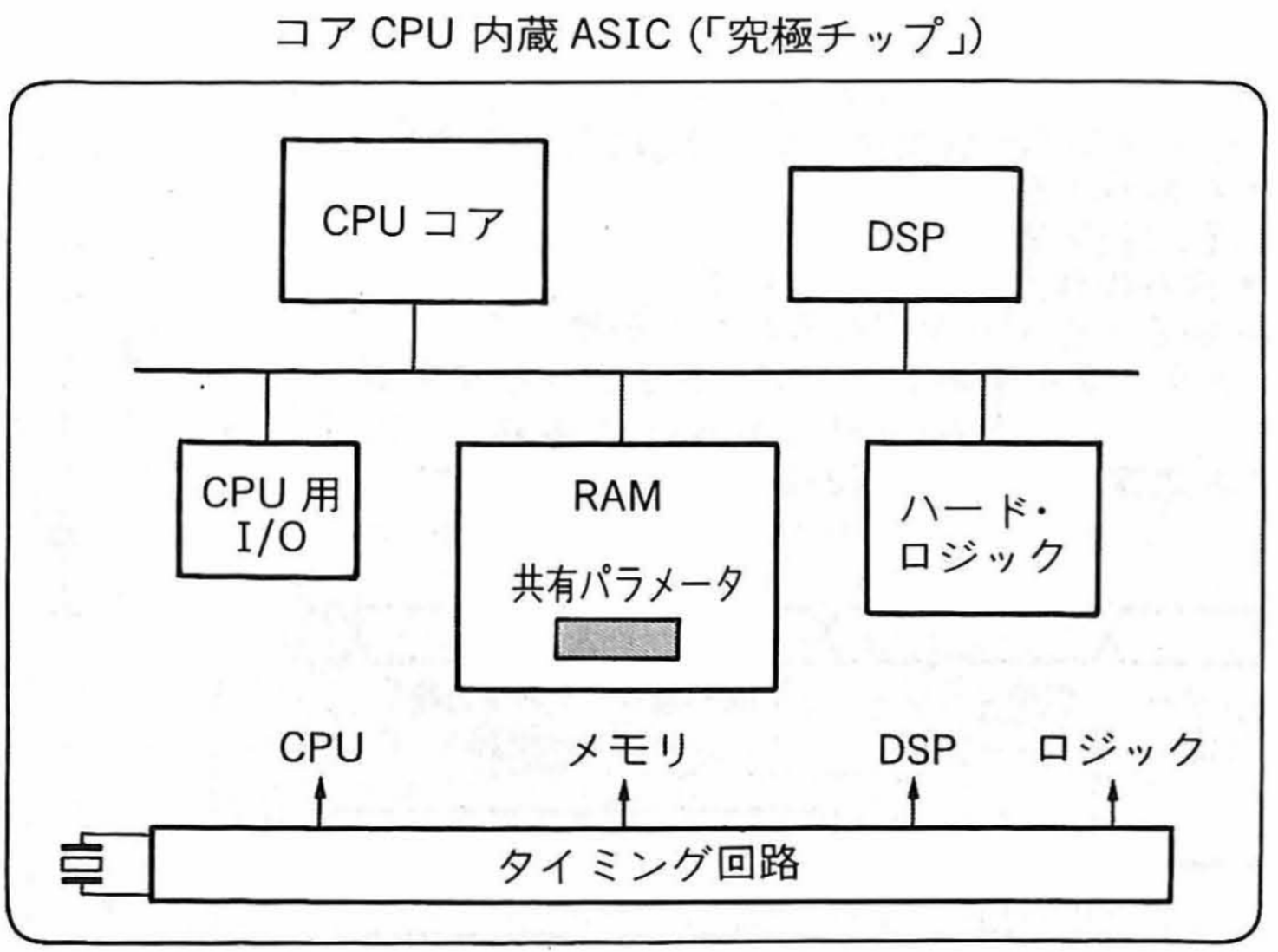


ノイマンアーキテクチャの呪縛との戦い

（文献1）

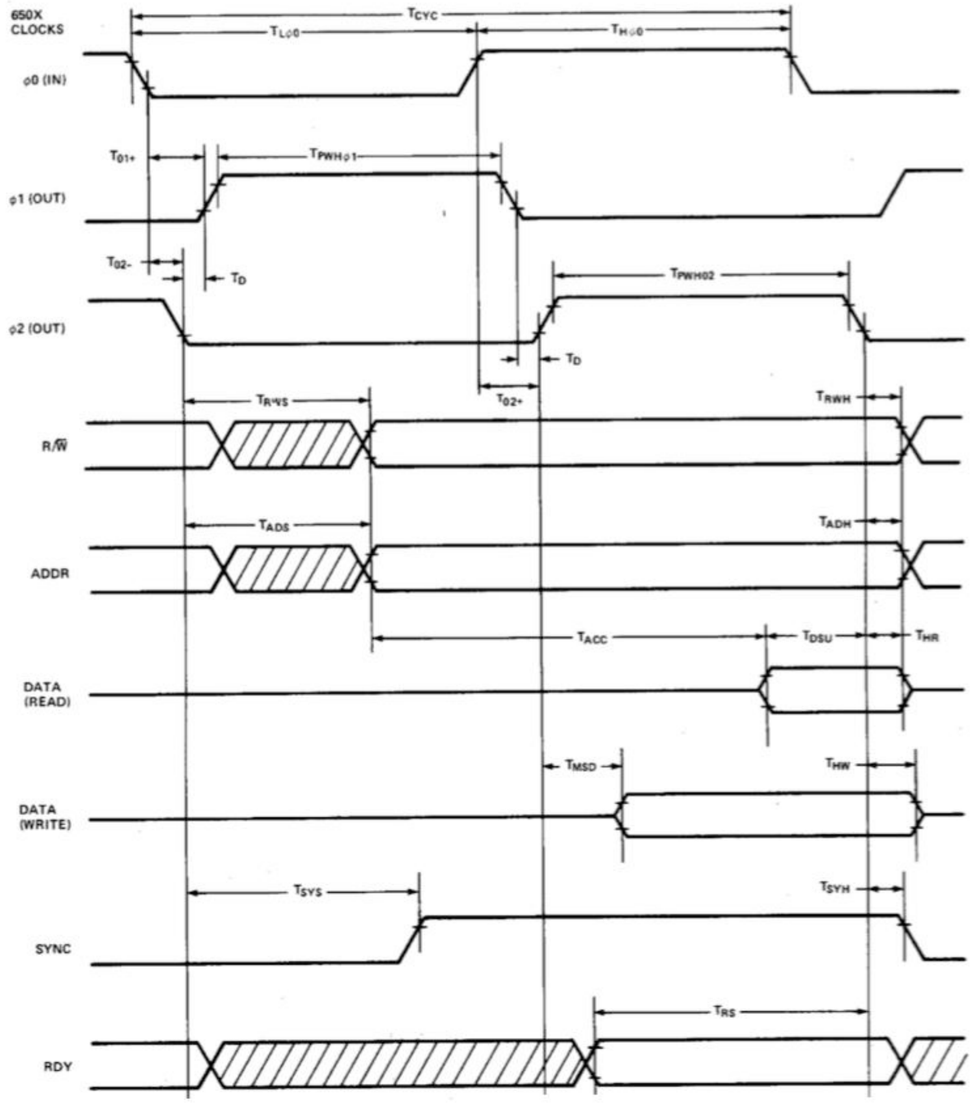


（文献1）



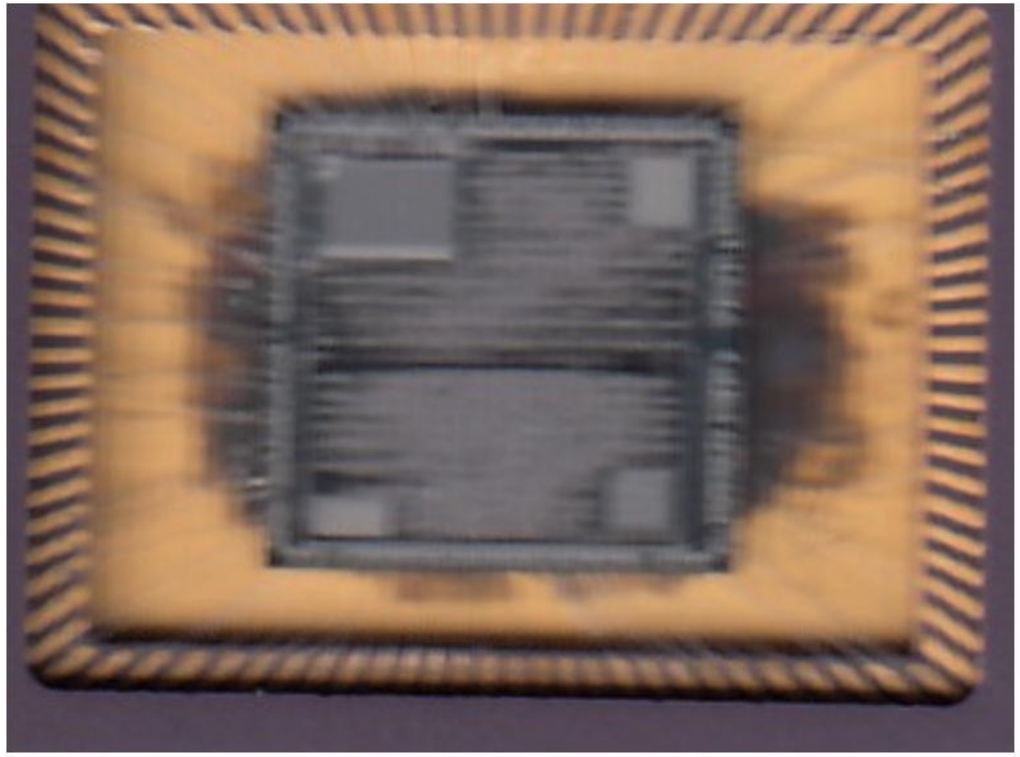

☆ チップ内の RAM に共有パラメータをもって CPU と DSP が交互にアクセスする

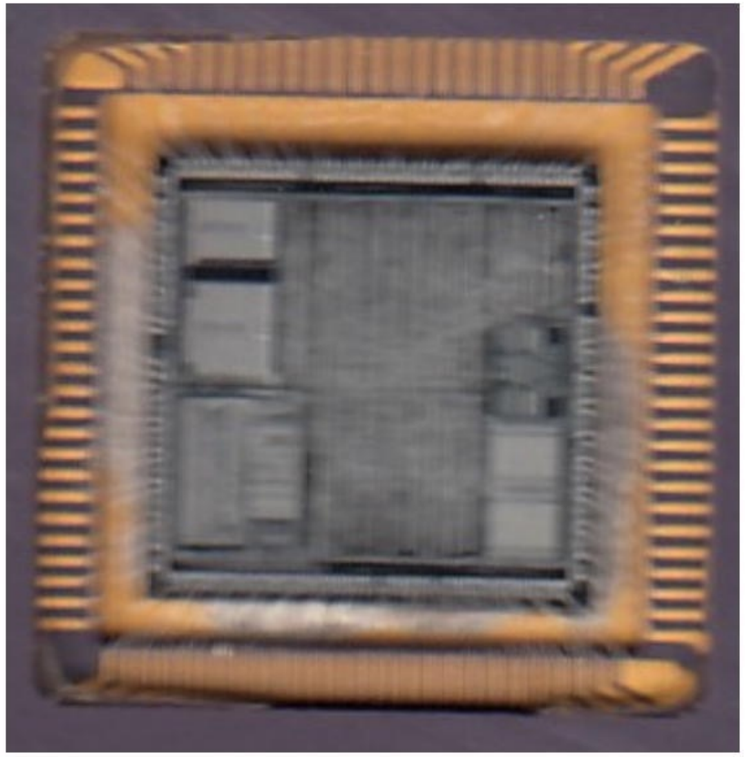
# ロックウェル6502 のCPUサイクル



「音源LSI」(1.5$\mu$m)

↓





↑

「音源LSI」+「6502CPU」(1.2$\mu$m)

# 基板上にCPUが無い!!

KAWAI
MS710


1992

# その効用

・ローエンド低速で高効率達成

・CPU/DSPのROM 2個→1個(共有)

・低消費電力

・業界騒然(基板上にCPUが無い!!)

→ご褒美で国際会議出張OK

# （ここだけの話）

# 本日の話題

(1)データフロー・プロセッサ

(2)Cycle-Steel手法

(3)Propellerプロセッサ

# 本日の話題

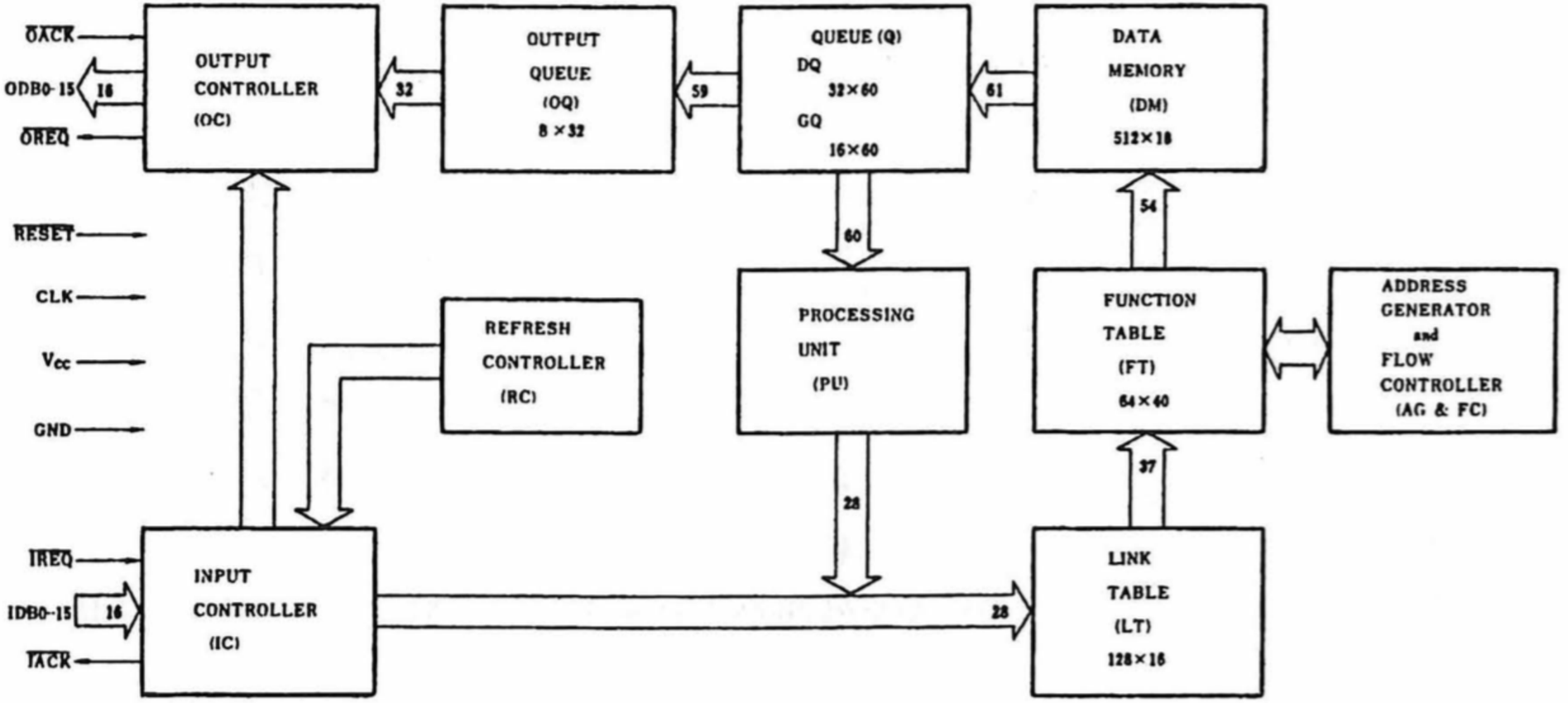
(1)データフロー・プロセッサ

(2)Cycle-Steel手法

(3)Propellerプロセッサ

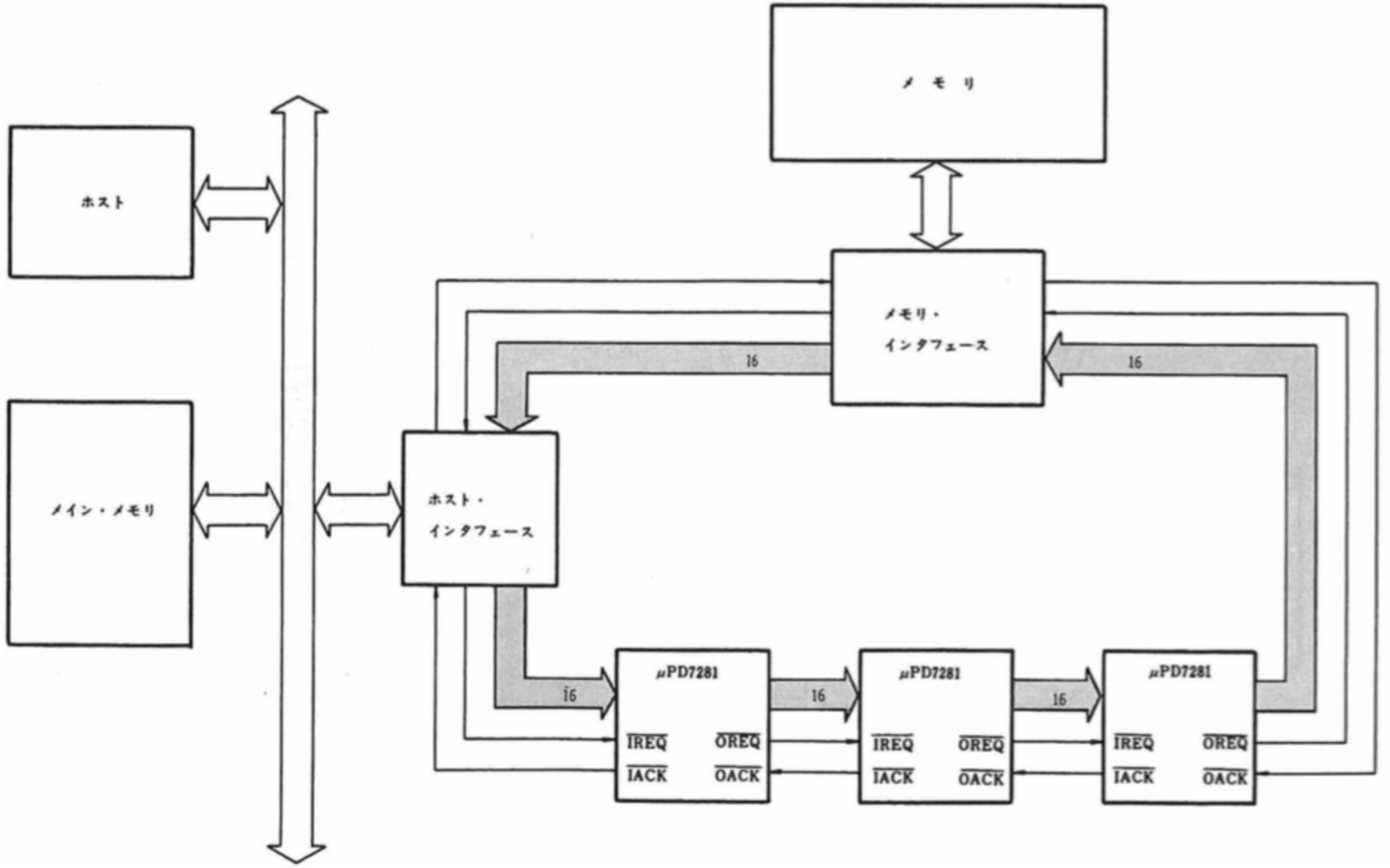
（文献1）

データフロー・プロセッサ ImPP(日本電気のデータブックより)



（文献1）

ImPP の並べ方はかんたん！（日本電気データブックより）



# その検討

・コマンドとデータとを一緒に扱う

・並列処理/画像処理に向いている

・ランダム要素には弱い

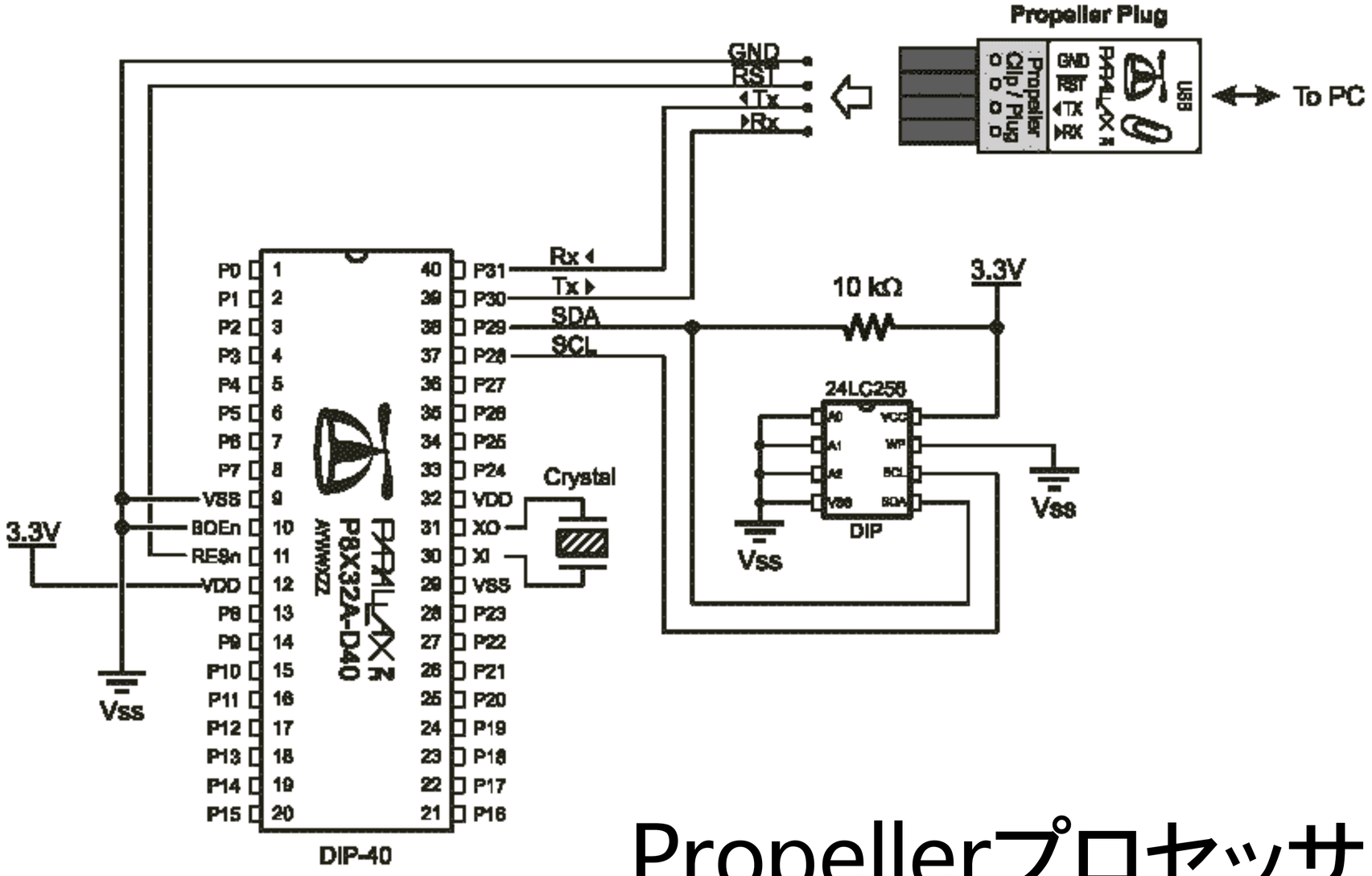
(楽器には向かない)

# 本日の話題

(1)データフロー・プロセッサ

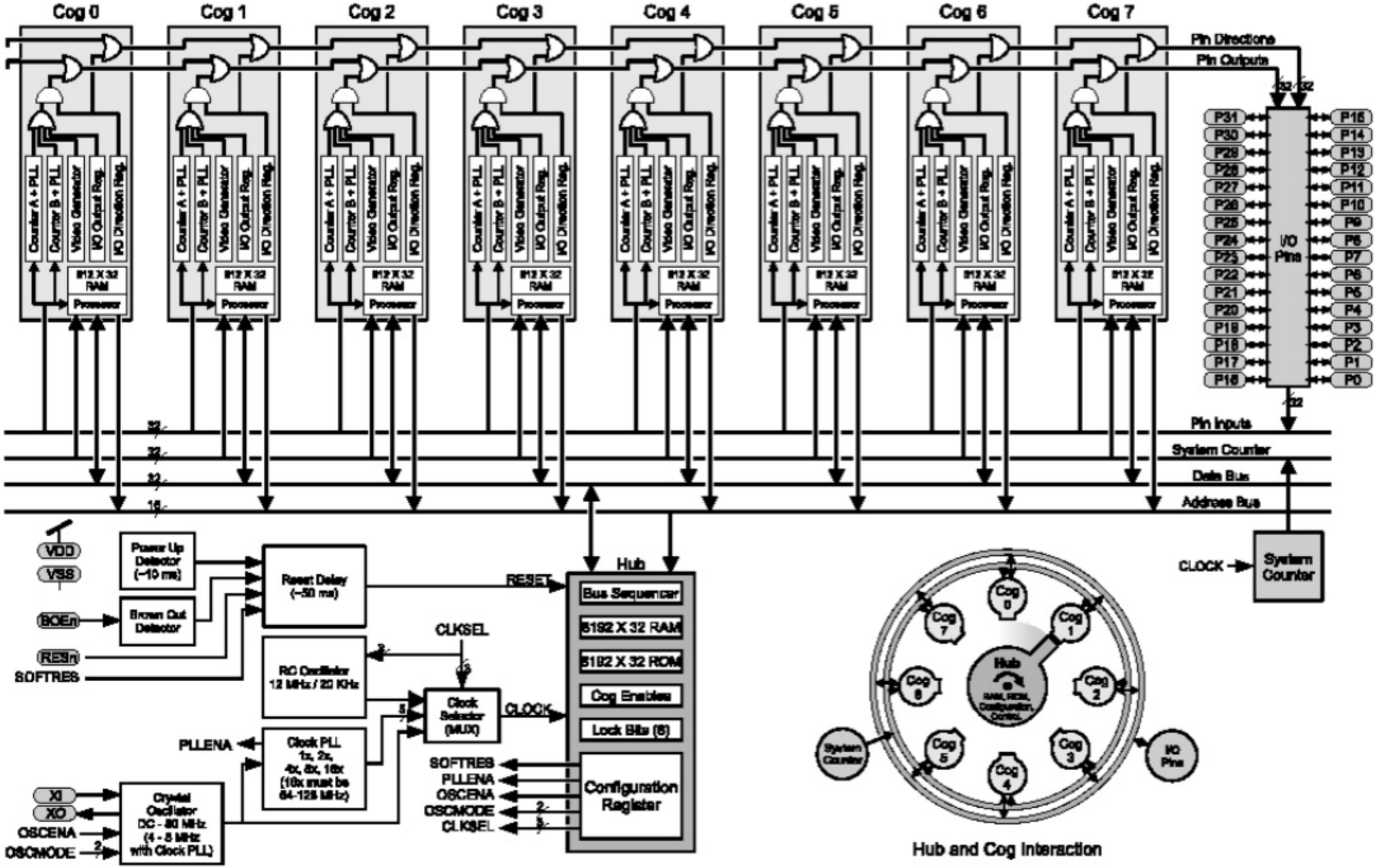
(2)Cycle-Steel手法

(3)Propellerプロセッサ

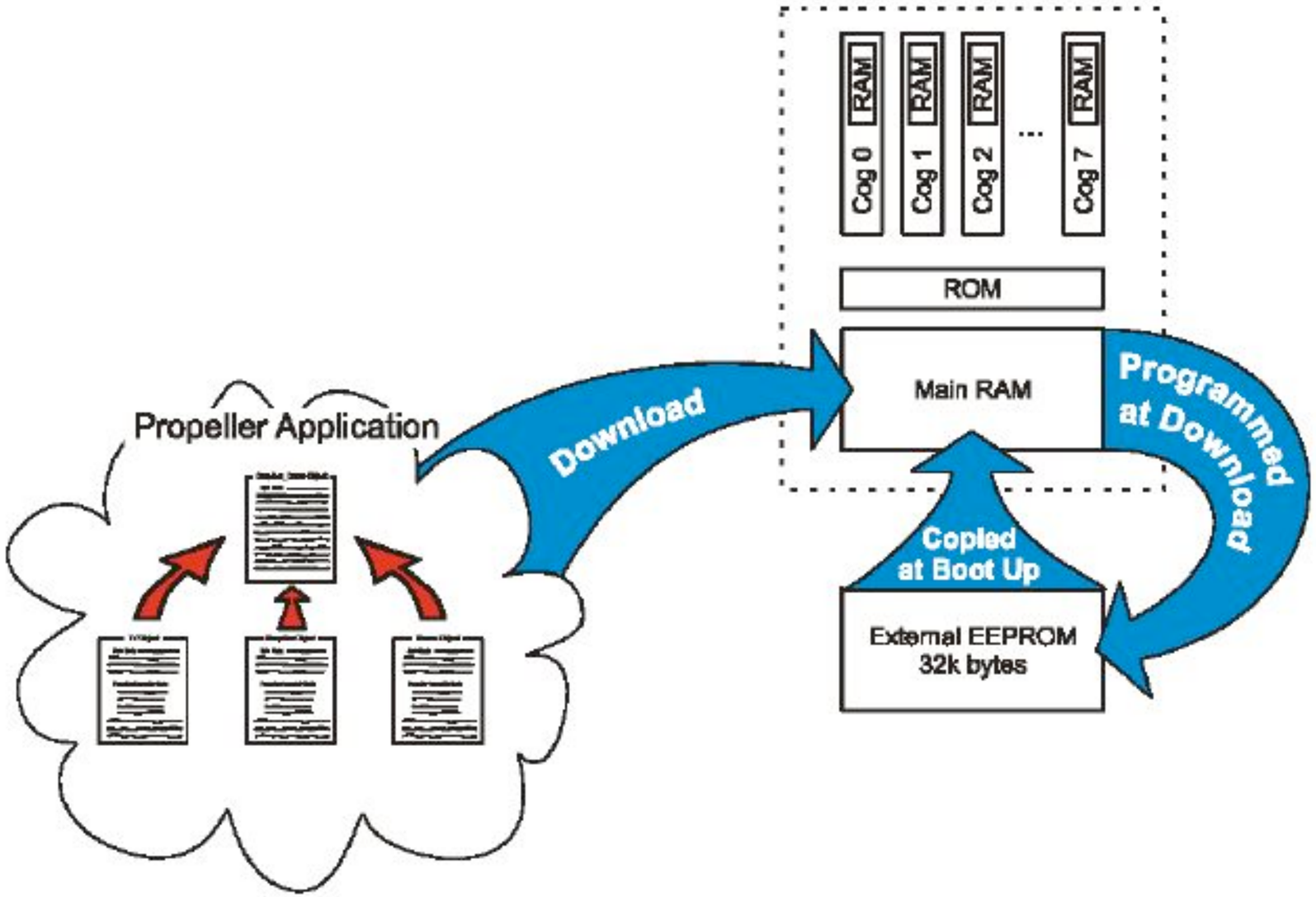



# Propellerプロセッサ

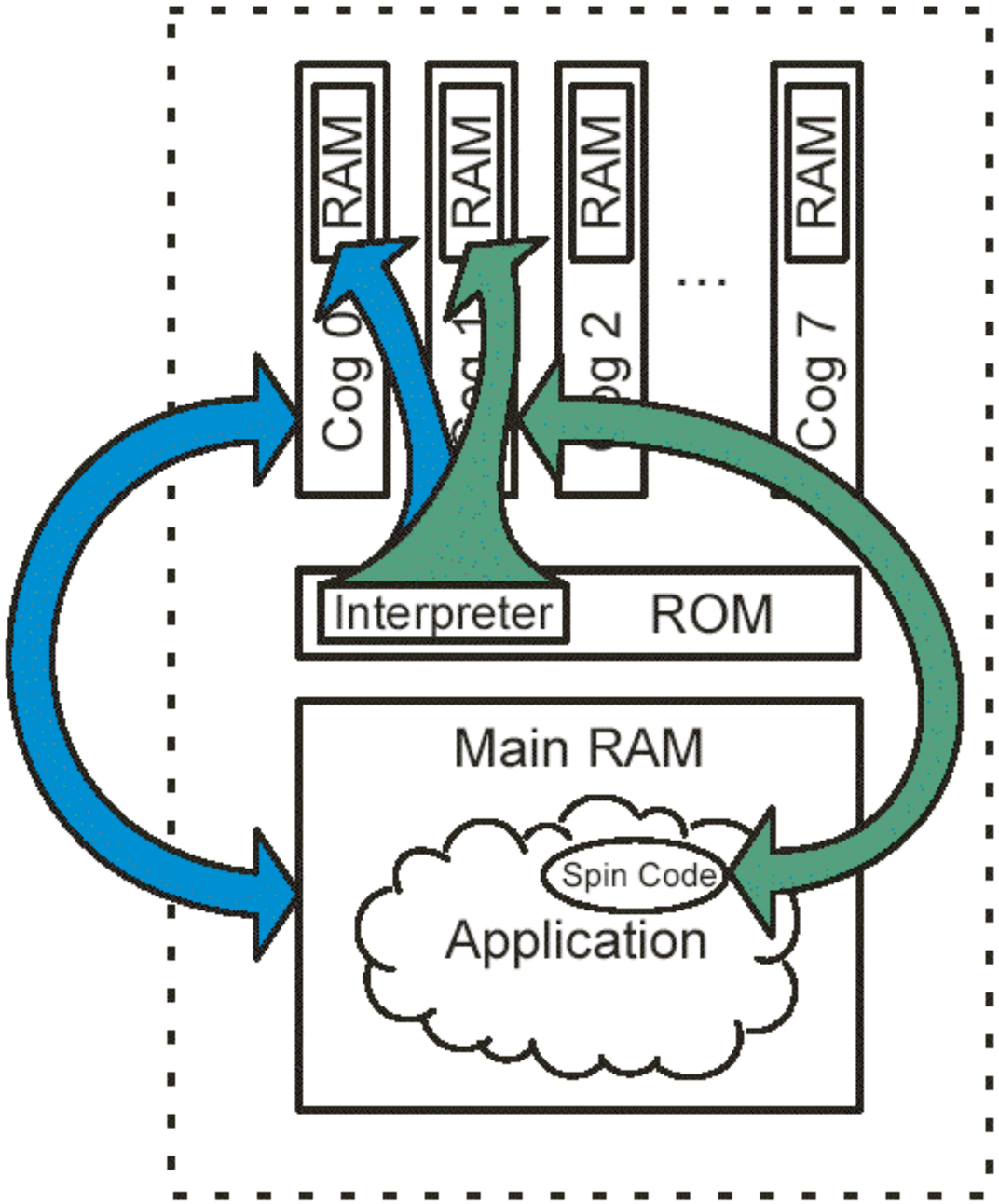
# ノイマンアーキテクチャの呪縛との戦い



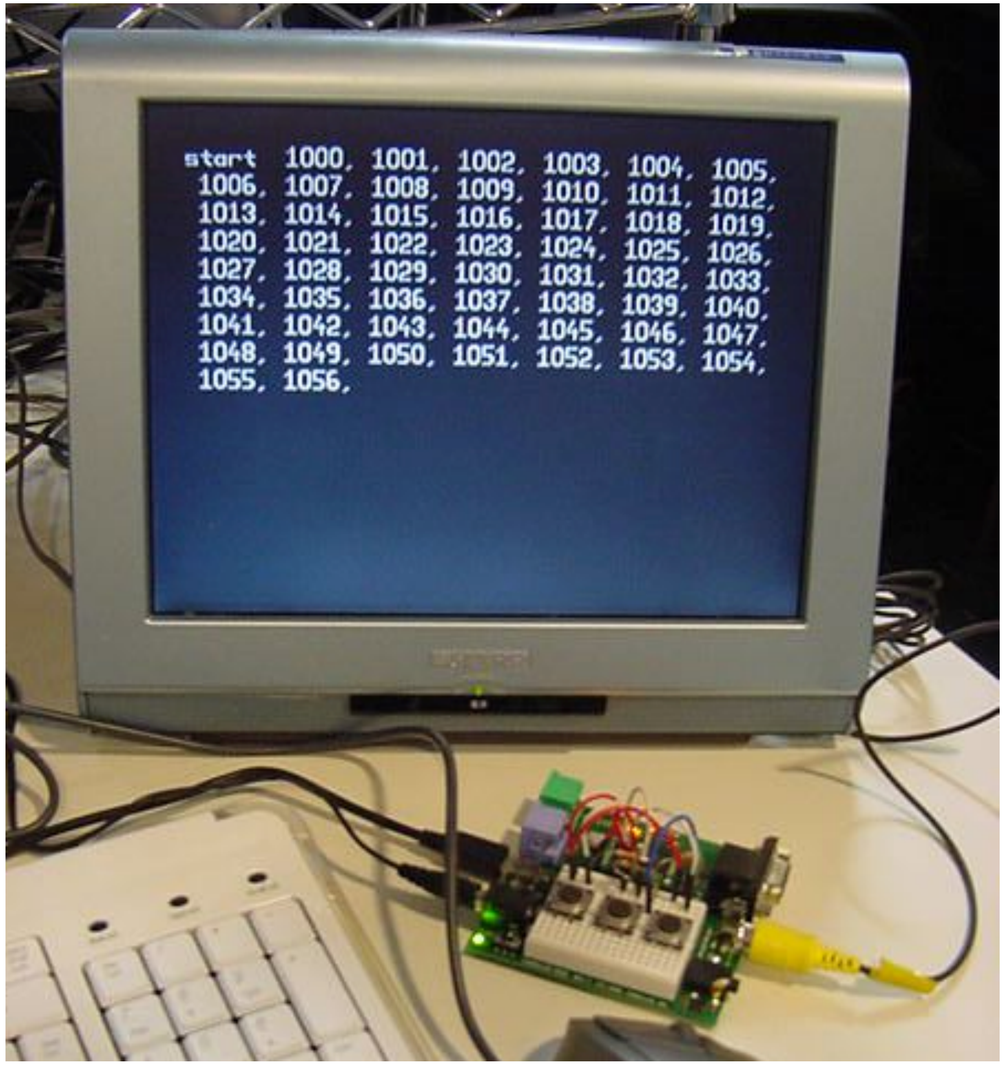
ノイマンアーキテクチャの呪縛との戦い

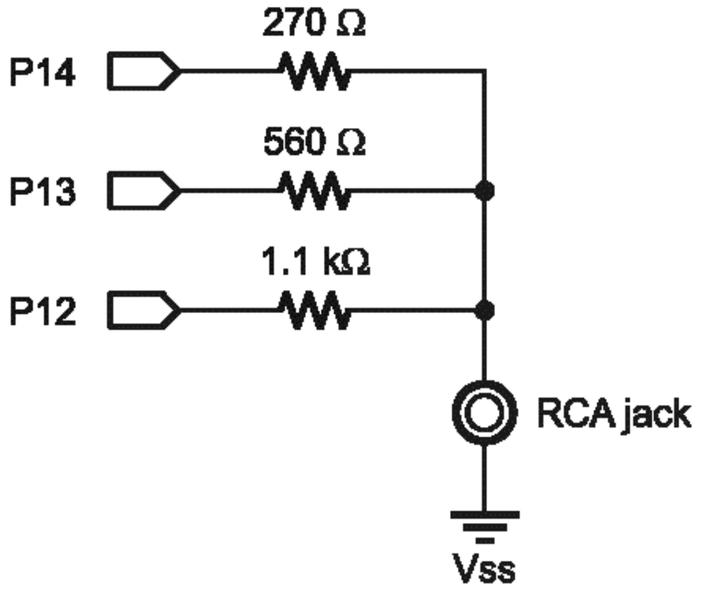


ノイマンアーキテクチャの呪縛との戦い

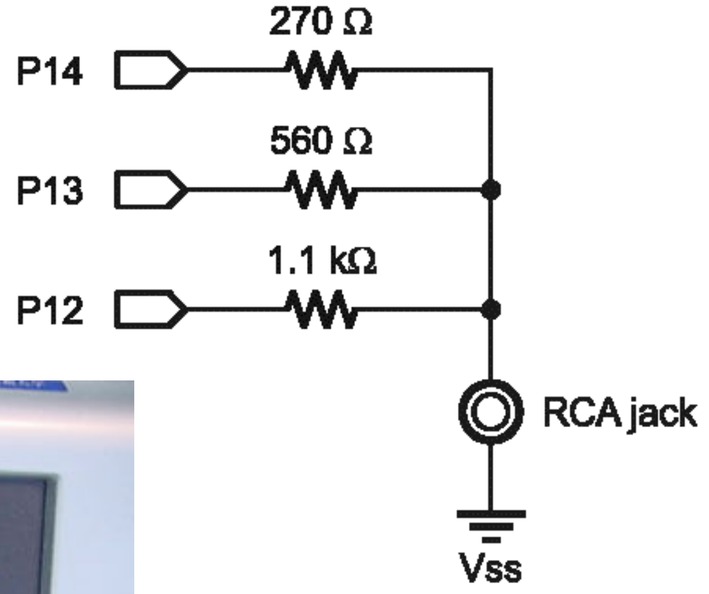


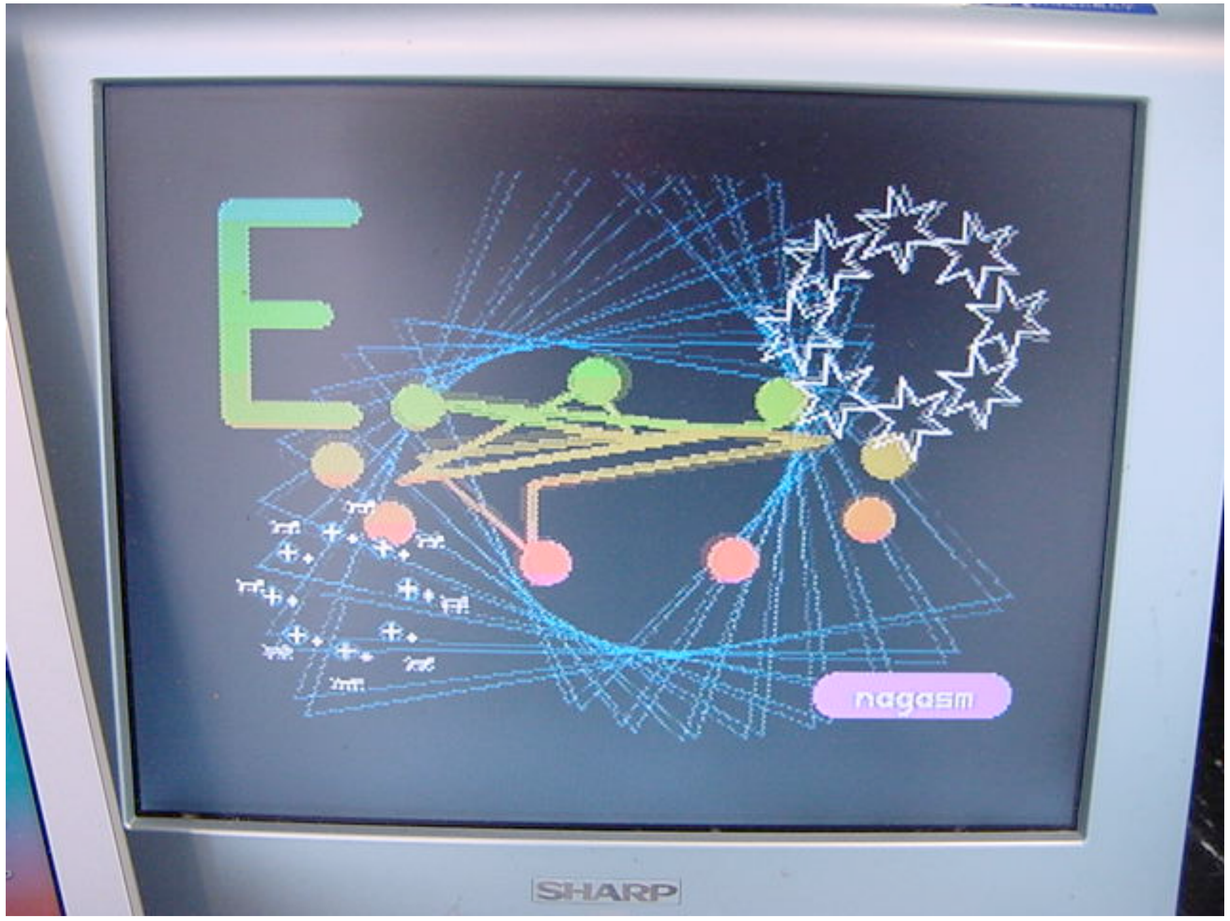
ノイマンアーキテクチャの呪縛との戦い



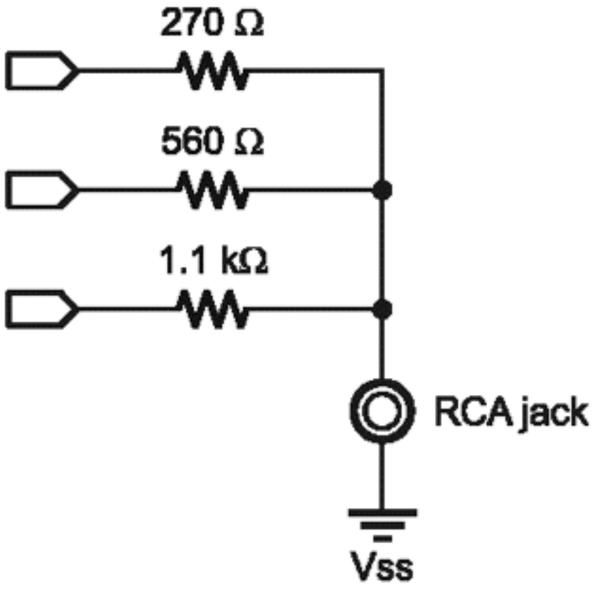



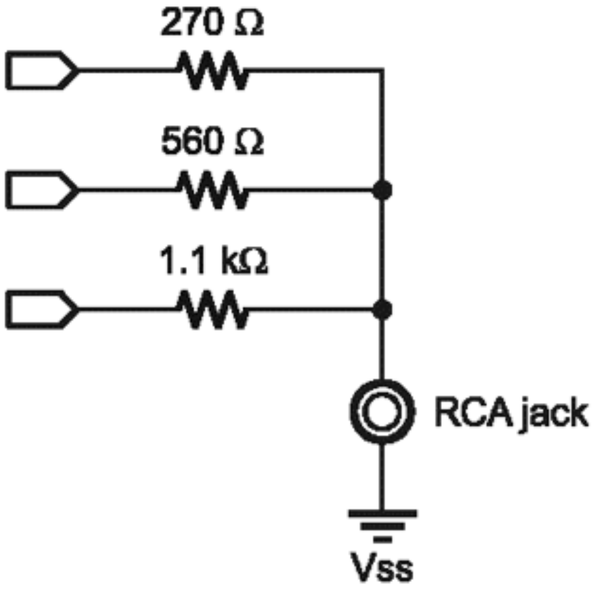
ノイマンアーキテクチャの呪縛との戦い

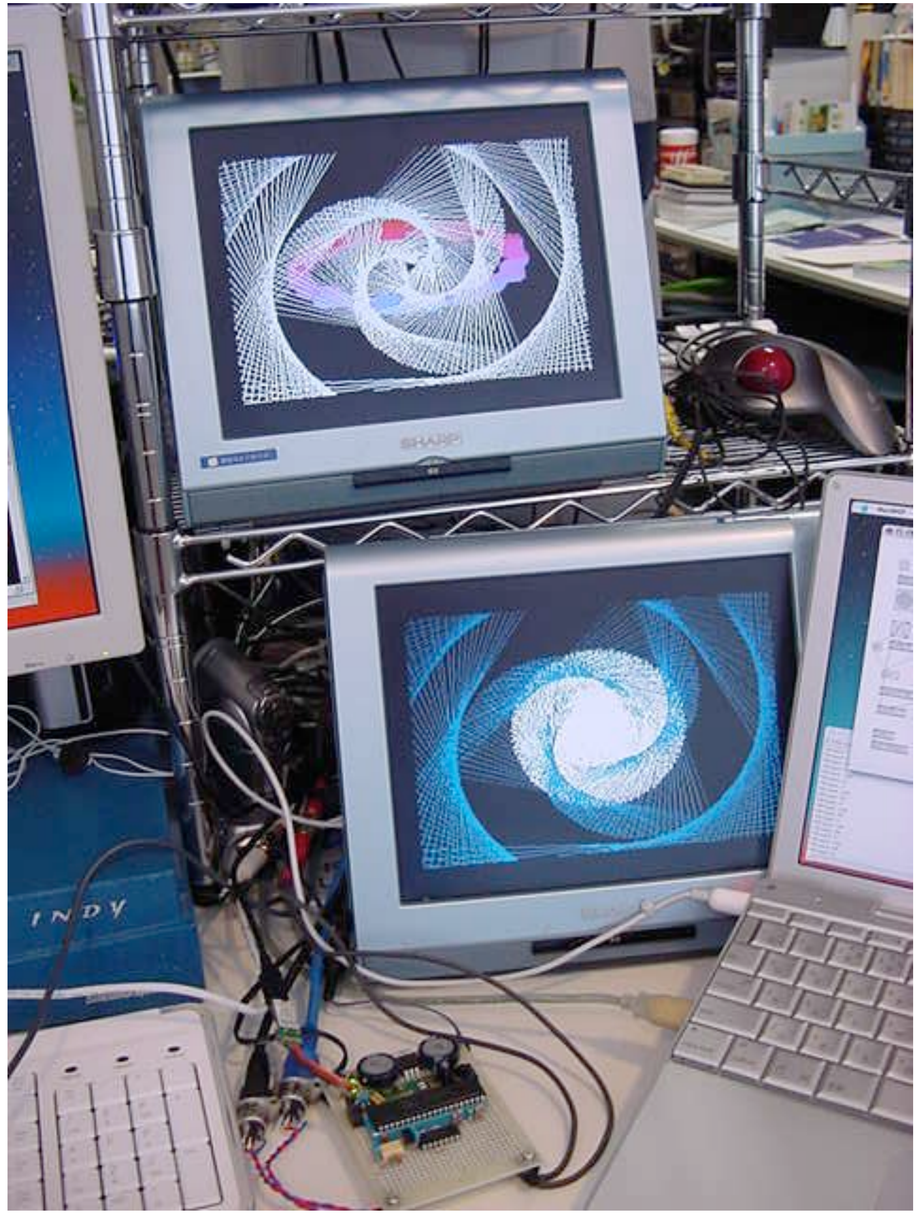


ノイマンアーキテクチャの呪縛との戦い

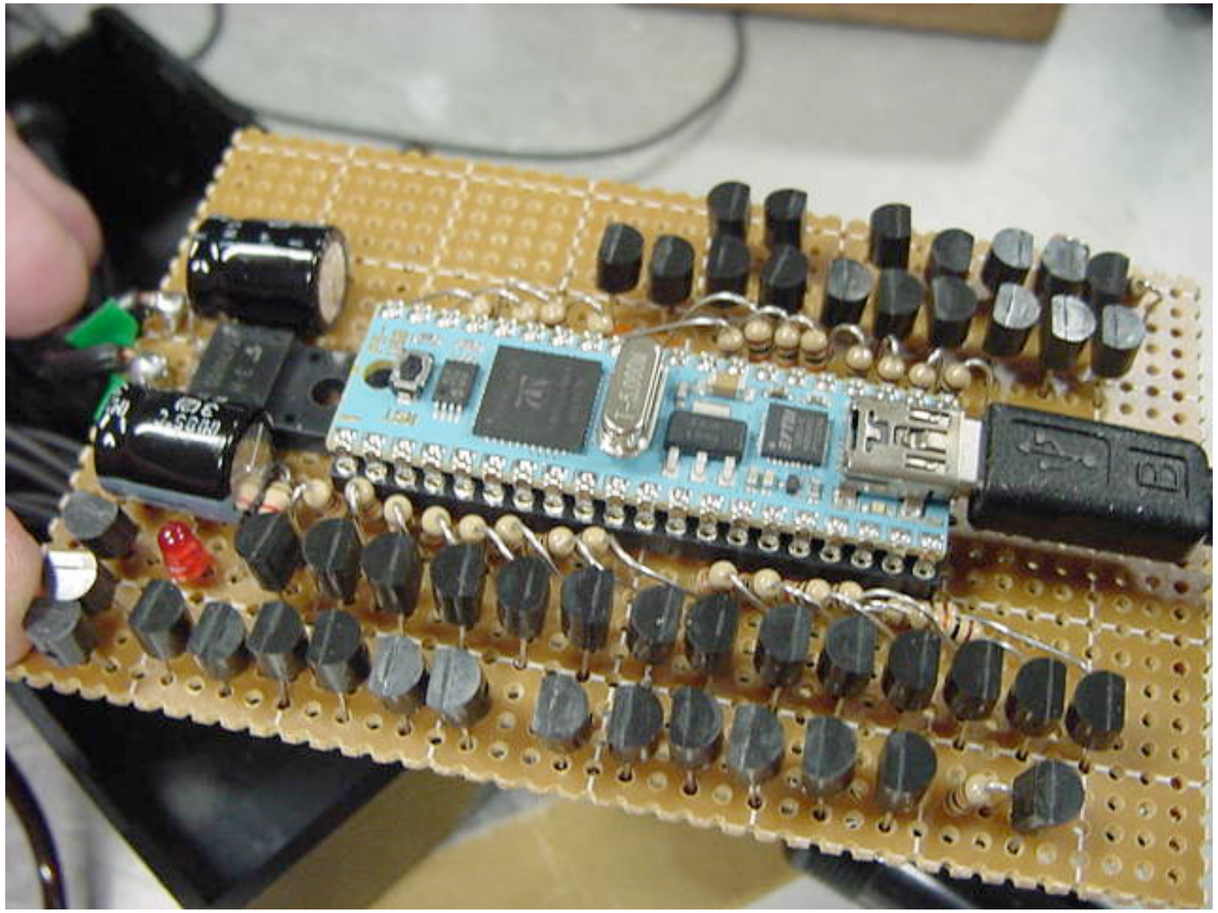


# Propeller活用作品

# Propeller活用作品

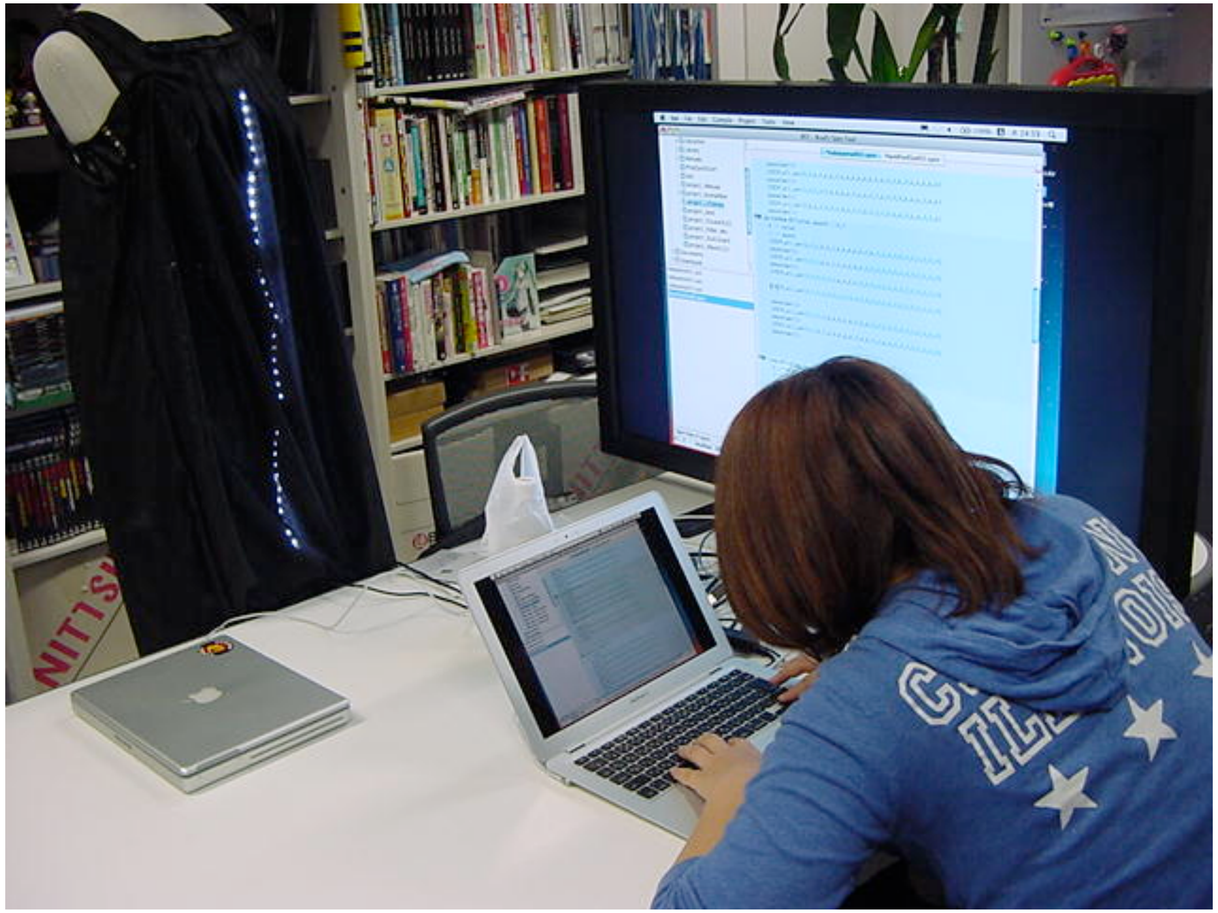


# Propeller活用作品

# Propeller活用作品

# Propeller活用作品

# Propeller活用作品



# Propeller活用作品

# どこが凄いのか

# RDLONG

**Instruction:** Read long of main memory.

RDLONG *Value*, ⟨#⟩ *Address*

---

**Result:** Long is stored in *Value*.

- ***Value*** (d-field) is the register to store the long value into.
- ***Address*** (s-field) is a register or a 9-bit literalwhose value is the main memory address to read from.

| –INSTR– ZCRI | –CON– | –DEST– | –SRC– | Z Result | C Result | Result | Clocks |
|---|---|---|---|---|---|---|---|
| 000010 001i | 1111 | ddddddddd | sssssssss | Result = 0 | --- | Written | 7..22 |

# JMPRET

**Instruction:** Jump to address with intention to “return” to another address.

JMPRET ***RetInstAddr***, ⟨#⟩ ***DestAddress***

**Result:** PC + 1 is written to the s-field of the register indicated by the d-field.

- ***RetInstAddr*** (d-field) is the register in which to store the return address (PC + 1); it should be the address of an appropriate **RET** or **JMP** instruction for *DestAddress*.
- ***DestAddress*** (s-field) is the register or 9-bit literal whose value is the address to jump to.

| –INSTR– ZCRI –CON– –DEST– –SRC– | Z Result | C Result | Result | Clocks |
|---|---|---|---|---|
| 010111 001i 1111 ddddddddd sssssssss | Result = 0 | --- | Written | 4 |

Cog内のRAMにロードされた
32ビットのコマンドの一部を上書き
(書き換え)して再利用することに
より驚異的なパフォーマンスを実現

# 詳しくは→「Propeller日記」

# That’s all, thank you.