「TW4」

れもん

自己紹介

- ID: lemoncmd
- 低レイヤ全般好き
- 自作と理解の繰り返し

れもん

↑残像

技術書典で本を出しました

- レトロゲームを知りたい人のための
  特権命令・割り込み概論

- CPUにどうやって特権命令や割り込み
  を乗っけるかを解説する本です

ここから買えます→

技術書典で本を出しました

- レトロゲームを知りたい人のための
  特権命令・割り込み 概論

- CPUにどうやって**特権命令** や**割り込み**
  を乗っけるかを解説する本です

ここから買えます→

特権命令

割り込み

4

特権命令

割り込み

4

もうお分かりですね？

「TW4」

「TW4」

Tokken Warikomi 4bitCPU

TD4に 特権 と 割り込み を載せました

- アドレス長は**そのまま4bit**
- 命令長は**そのまま8bit**
- ナンもカンも互換性アリ☑。

このラーメンタイマーも

そのまま動きます。→

```
    out 0111    // LEDを3つ点灯
loop1:
    add a, 0001 // 16回ループ
    jnc loop1
loop2:
    add a, 0001 // また16回ループ
    jnc loop2
    out 0110    // LEDを2つ点灯
loop3:
    add a, 0001 // 16回ループ
    jnc loop3
loop4:
    add a, 0001 // また16回ループ
    jnc loop4
loop5:
    out 0000    // LEDを滅
    out 0100    // LEDを点
    add a, 0001 // そして16回ループ
    jnc loop5
    out 1000    // 端のLEDを点灯する
    jmp 1111    // おわり（無限ループ）
```

機能紹介

- 4つのCPUモード
-

機能紹介

- 4つのCPUモード
- 3つの追加命令
-

機能紹介

- 4つのCPUモード
- 3つの追加命令
- 2種類の割り込み
-

機能紹介

- 4つのCPUモード
- 3つの追加命令
- 2種類の割り込み
- あと例外
-

機能紹介

- 4つのCPUモード
- 3つの追加命令
- 2種類の割り込み
- あと例外
- あと仮想メモリ
-

機能紹介

- 4つのCPUモード
- 3つの追加命令
- 2種類の割り込み
- あと例外
- あと仮想メモリ
- あとシステムレジスタ
-

機能紹介

- 4つのCPUモード
- 3つの追加命令
- 2種類の割り込み
- あと例外
- あと仮想メモリ
- あとシステムレジスタ
- あと色々

4つのCPUモード

- ユーザーモード
- ソフトウェア割り込みモード
- 例外モード
- ハードウェア割り込みモード

} 特権モード

ちょっと待って？

どうやって互換性を保ちながら割り込みハンドラを？

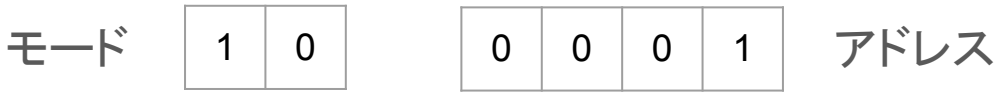
- 16個しか命令を入れられないはず
- でもラーメンタイマーも動くって言ってたし……
- どこに割り込みハンドラを詰め込むの？

どうやって互換性を保ちながら割り込みハンドラを？

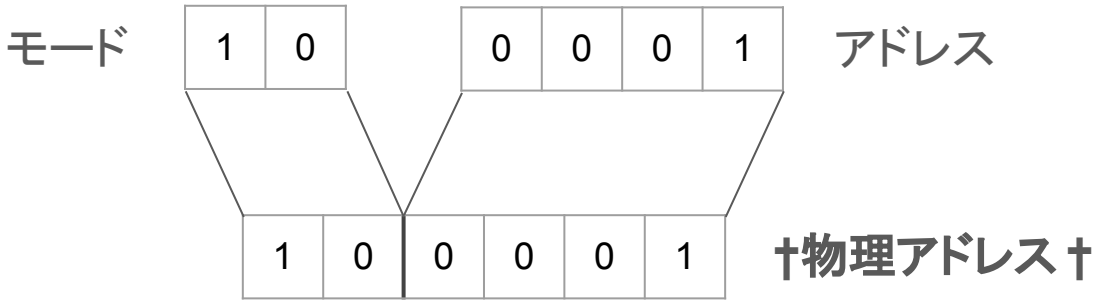
- 16個しか命令を入れられないはず
- でもラーメンタイマーも動くって言ってたし……
- どこに割り込みハンドラを詰め込むの？

→仮想メモリ空間を使います

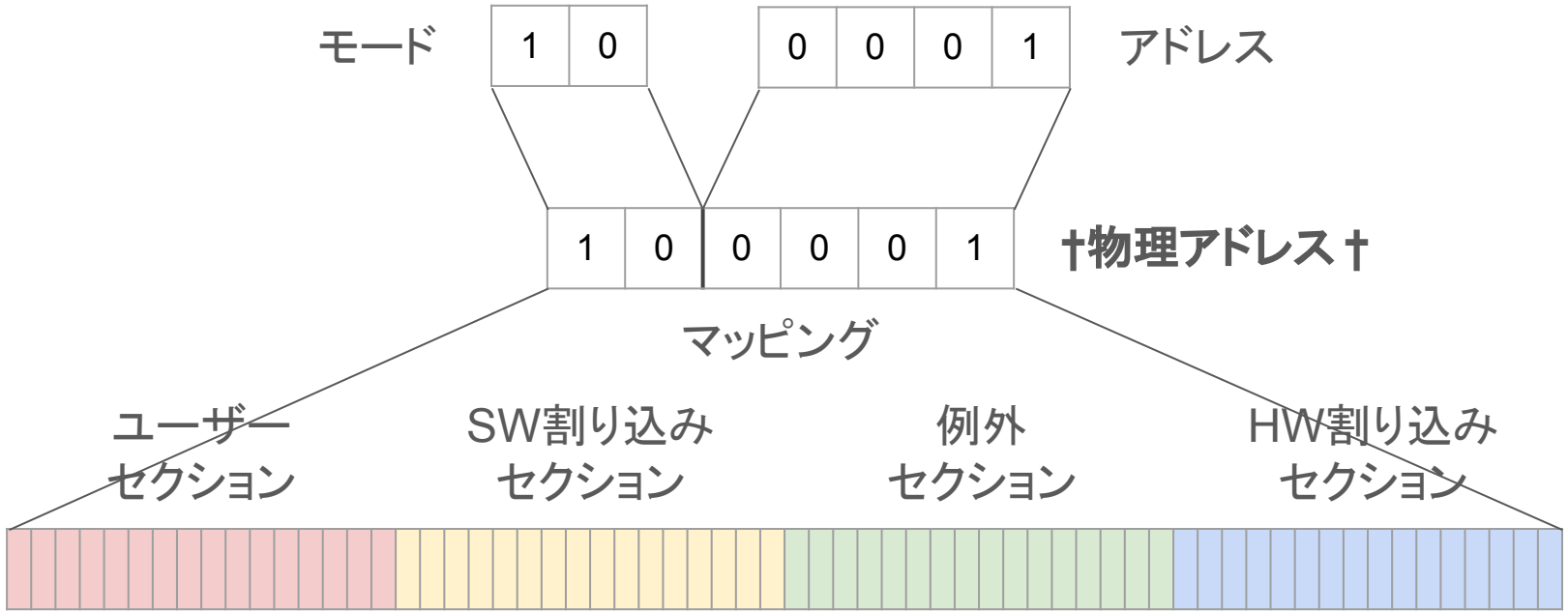
仮想メモリ（？）



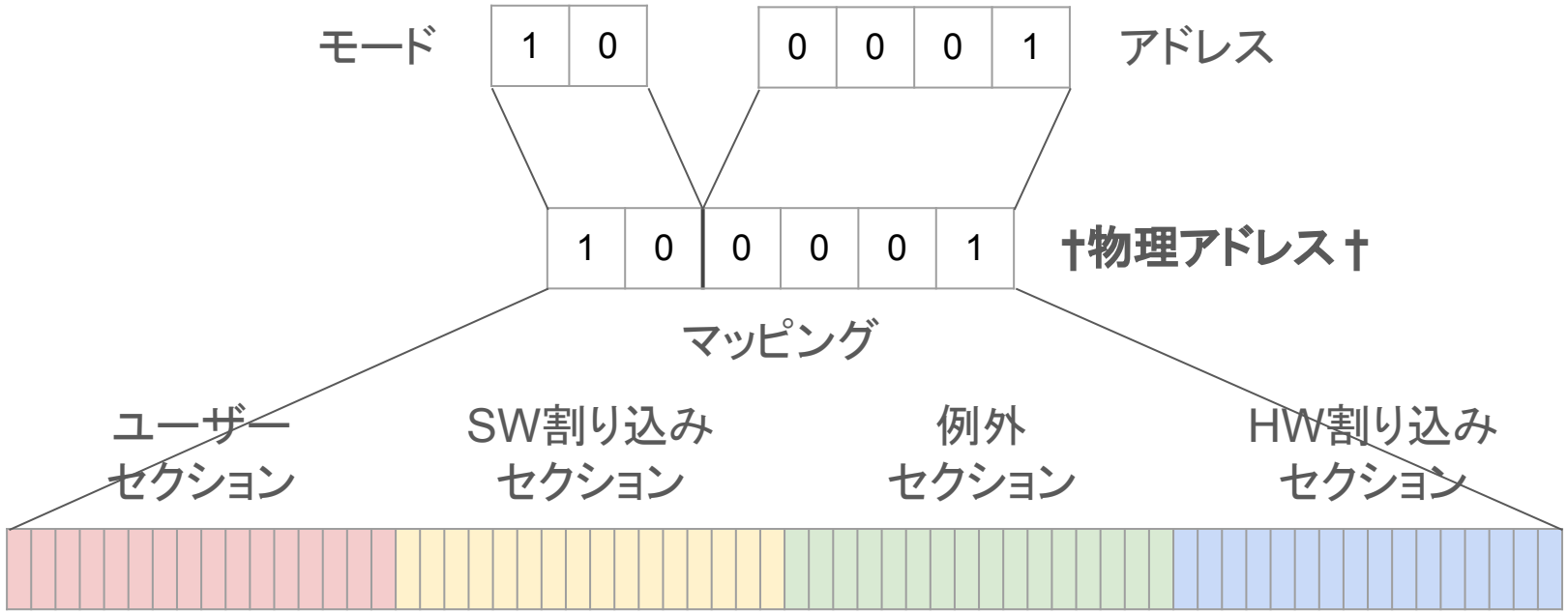
仮想メモリ（？）



仮想メモリ(？)


モード
1
0
0
0
0
1
アドレス
1
0
0
0
0
1
†物理アドレス†
マッピング
ユーザー
セクション
SW割り込み
セクション
例外
セクション
HW割り込み
セクション

仮想メモリ(？)



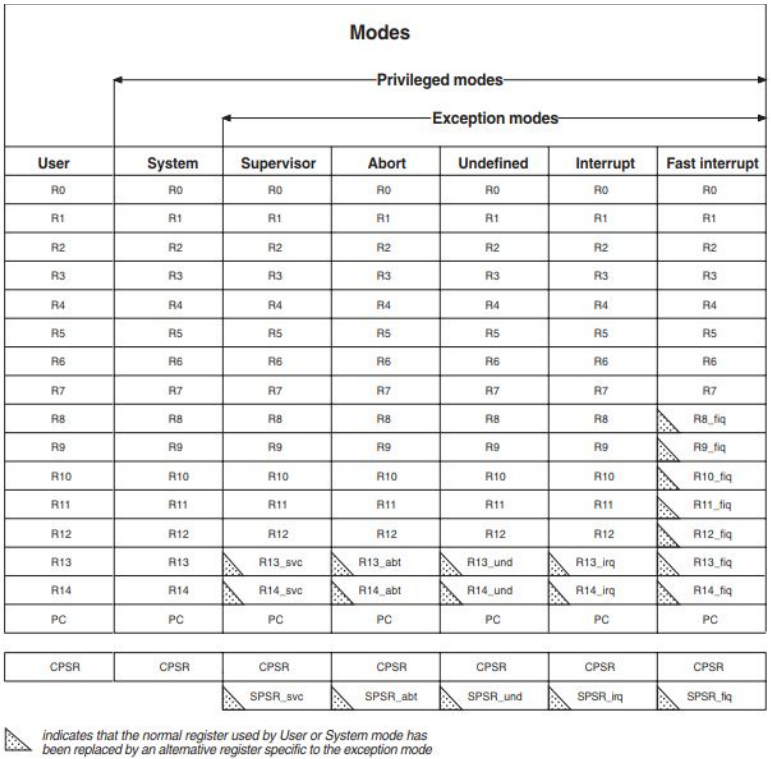
↑の表を作るのに**44回右クリック**しました

レジスタの退避は？

- 割り込み処理しちゃったらユーザーモードのレジスタ壊しちゃいそう
- でもTD4って退避する用のRAMとかないよね……？

参考：ARMのレジスタ

- CPUのモードごとに独立した汎用レジスタがある（↓はARMv4～ARMv6）



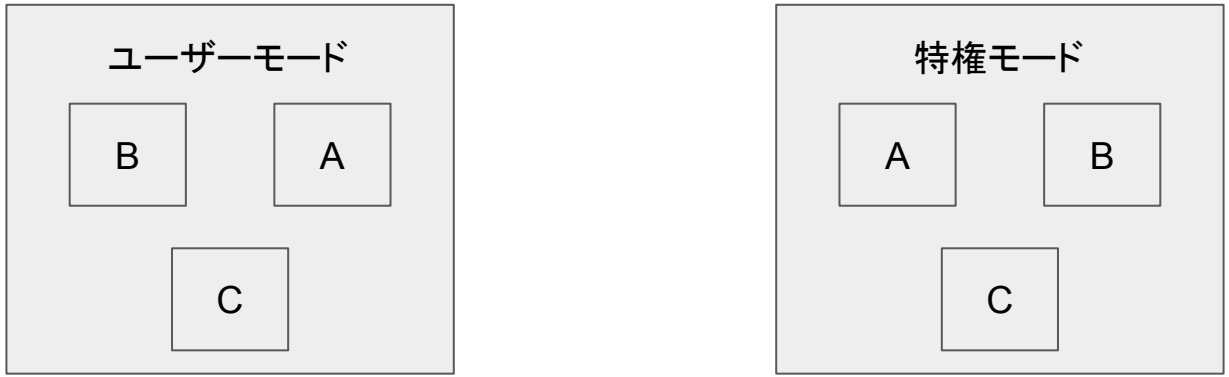
Modes						
	Privileged modes					
		Exception modes				
User	System	Supervisor	Abort	Undefined	Interrupt	Fast interrupt
R0	R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7	R7
R8	R8	R8	R8	R8	R8	R8_fiq
R9	R9	R9	R9	R9	R9	R9_fiq
R10	R10	R10	R10	R10	R10	R10_fiq
R11	R11	R11	R11	R11	R11	R11_fiq
R12	R12	R12	R12	R12	R12	R12_fiq
R13	R13	R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
R14	R14	R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
PC	PC	PC	PC	PC	PC	PC
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
		SPSR_svc	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

indicates that the normal register used by User or System mode has been replaced by an alternative register specific to the exception mode

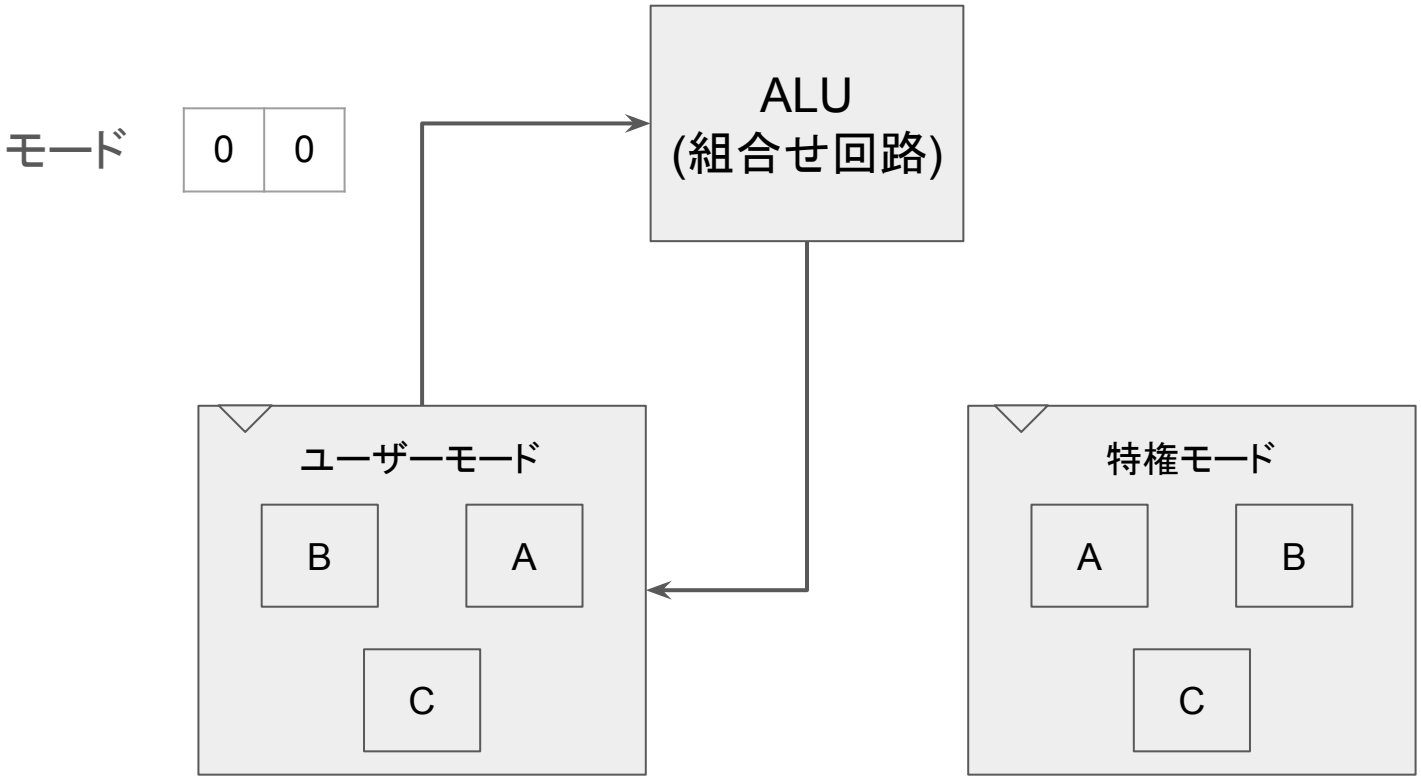
レジスタの退避は？

- 割り込み処理しちゃったらユーザーモードのレジスタ壊しちゃいそう
- でもTD4って退避する用のRAMとかないよね……？

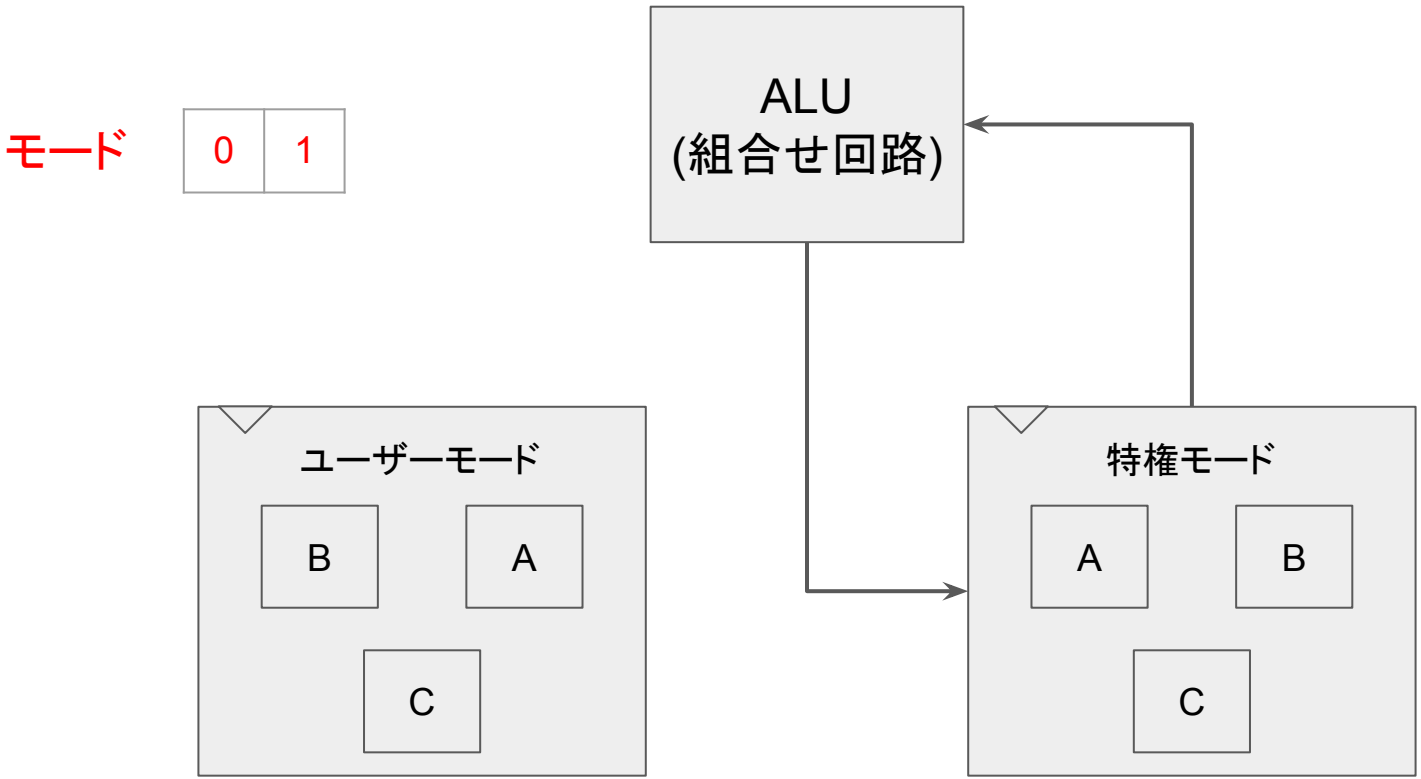
- 特権モード用にレジスタ作ろう！



レジスタの切り替え



レジスタの切り替え



レジスタの入れ替え

モード

0	1

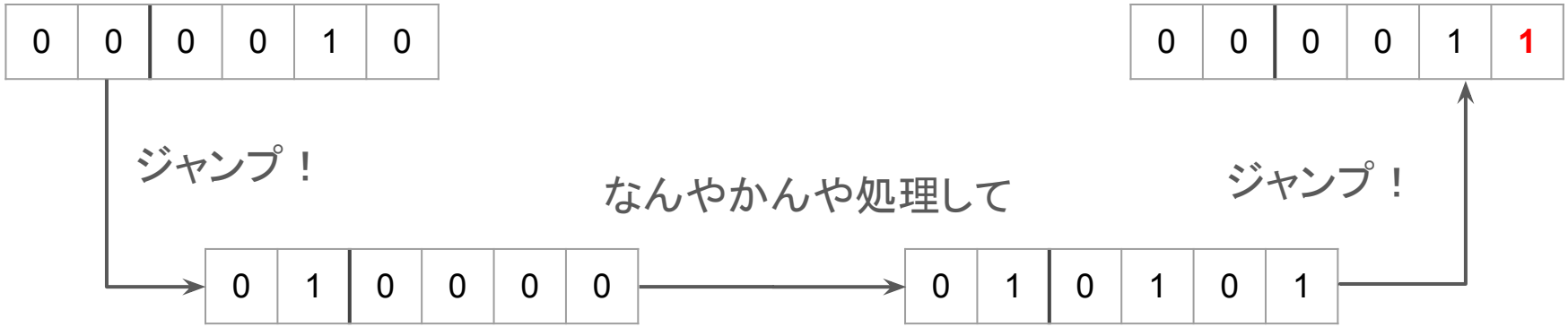
ALU
(組合せ回路)

ソフトウェア割り込み

- ハンドラ内で特権が使えるだけのただのAPI

ソフトウェア割り込みの実装

- SWI命令はモードとインストラクションポインタ切り替えるだけです
  - ただのジャンプとほぼほぼ変わらない
- IRET命令で戻る時用にインストラクションポインタ＋1は保存しておきましょう



例外

- 命令の実行中に何か**異常事態**が発生した際に例外ハンドラ起動
- **異常事態**って何があるか考えてみよう！
  -

例外

- 命令の実行中に何か**異常事態**が発生した際に例外ハンドラ起動
- **異常事態**って何があるか考えてみよう！
  - 未使用のオペランド（TD4だと1000, 1010, 1100, 1101が未使用）
  -

例外

- 命令の実行中に何か**異常事態**が発生した際に例外ハンドラ起動
- **異常事態**って何があるか考えてみよう！
  - 未使用のオペランド（TD4だと1000, 1010, 1100, 1101が未使用）
  - 即値が使われていない命令で、即値が 0000以外だった場合（OUT Bなど）

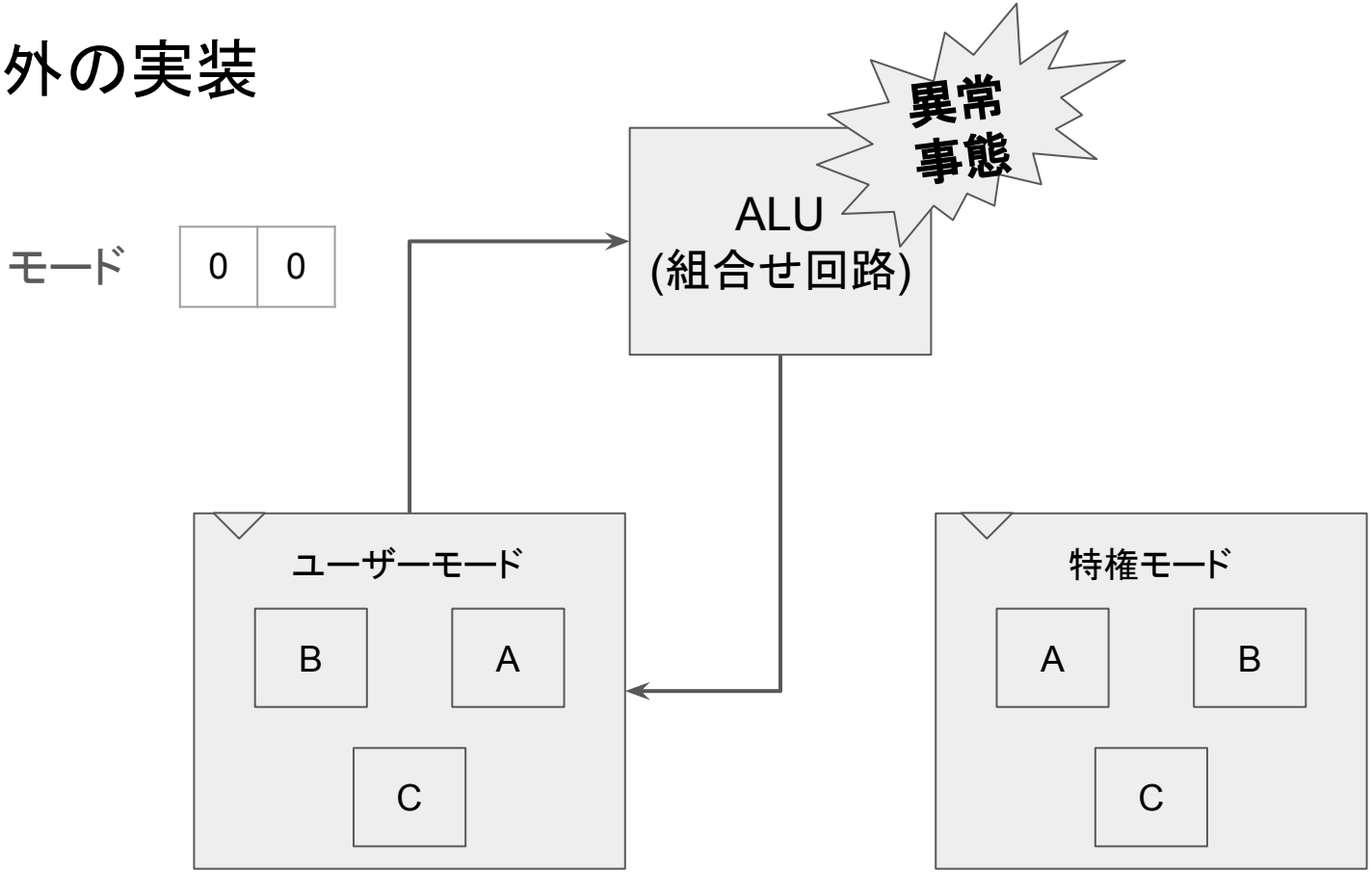
例外

- 命令の実行中に何か**異常事態**が発生した際に例外ハンドラ起動
- **異常事態**って何があるか考えてみよう！
  - 未使用のオペランド（TD4だと1000, 1010, 1100, 1101が未使用）
  - 即値が使われていない命令で、即値が 0000以外だった場合（OUT Bなど）
  - **16番目の命令から次に進んでしまった場合**
    - TD4実装では最初の命令に戻ってしまう……これは本当に意図した挙動？

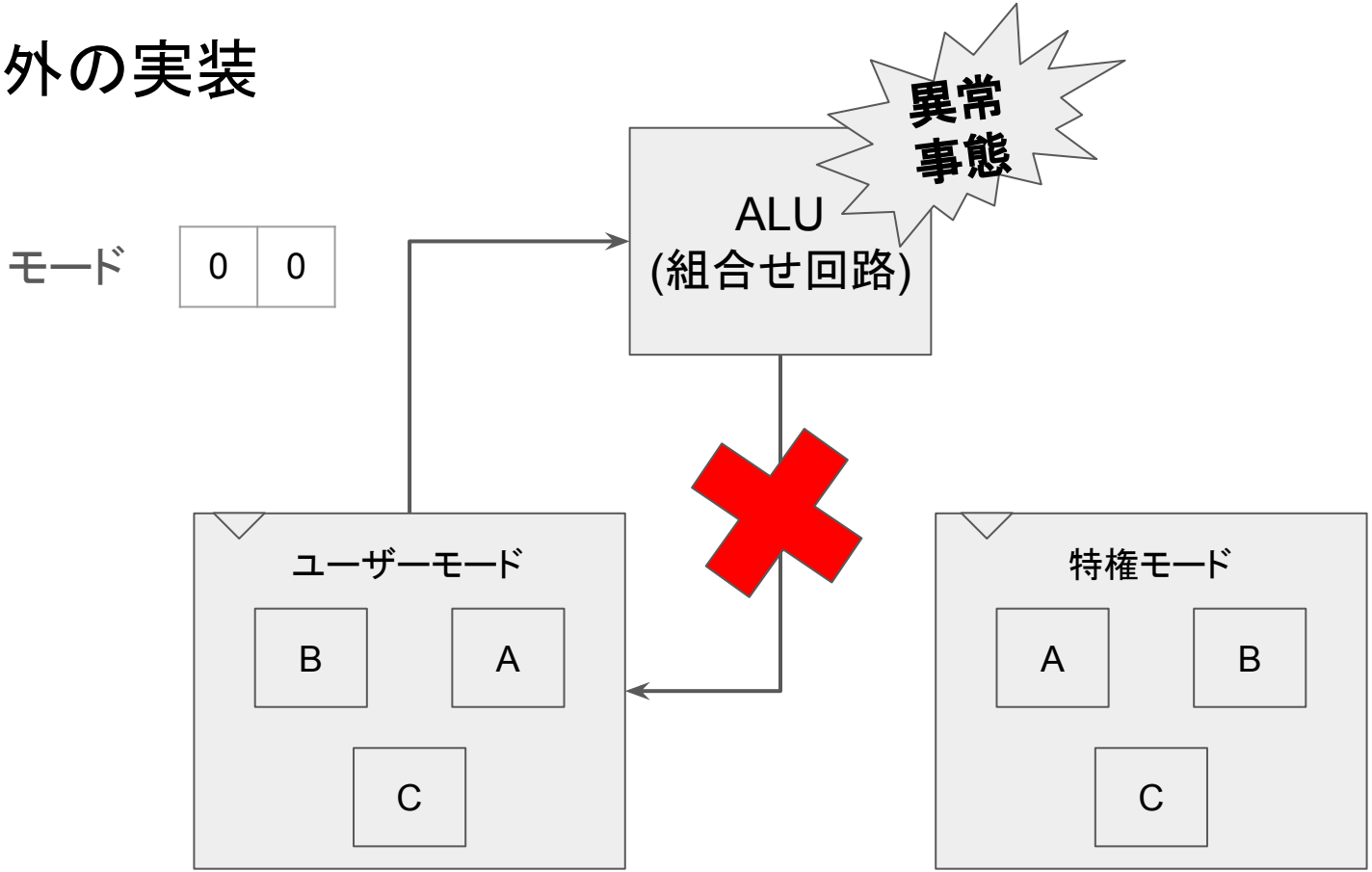
例外の実装

- 異常事態が発生した命令を**実行しなかった**ことにしたいね🌾

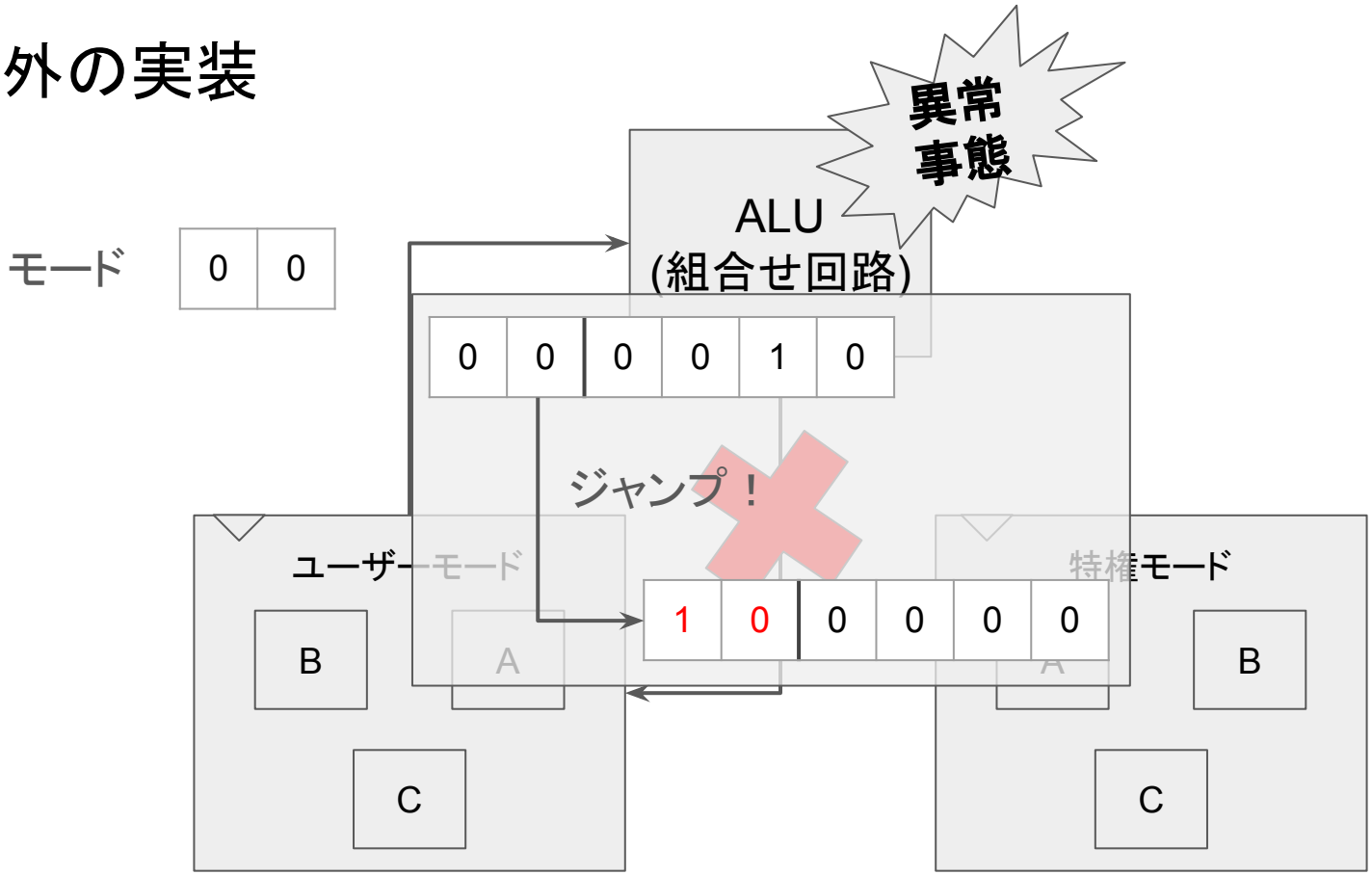
例外の実装



例外の実装



例外の実装


異常
事態
ALU
(組合せ回路)
モード
0 0
0 0 0 0 1 0
ジャンプ！
ユーザーモード
特権モード
1 0 0 0 0 0
B
A
C
B
A
C

例外中の例外

- ところで例外ハンドラの中で例外が起こったらどうなるの？

例外中の例外

- ところで例外ハンドラの中で例外が起こったらどうなるの？
- **ダブルフォールト！**
  - 完全にCPUの動作を停止するよう実装をします

ハードウェア割り込み

- 先に謝らないといけないのが、実は本の方で非同期のハードウェア割り込み対応できていません。

ハードウェア割り込み

- 先に謝らないといけないのが、実は本の方で非同期のハードウェア割り込み対応できていません。
- というわけで今回は非同期版について話します

ハードウェア割り込み

- 先に謝らないといけないのが、実は本の方で非同期のハードウェア割り込み対応できていません。
- というわけで今回は**非同期版**について話します

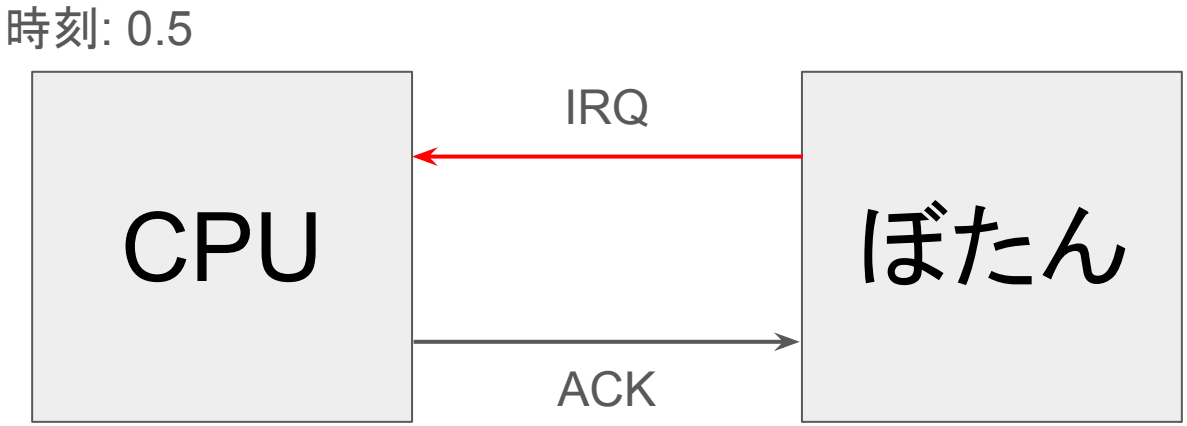
ハードウェア割り込み用ハードウェア

- 単純なのでボタンを使いましょう
-

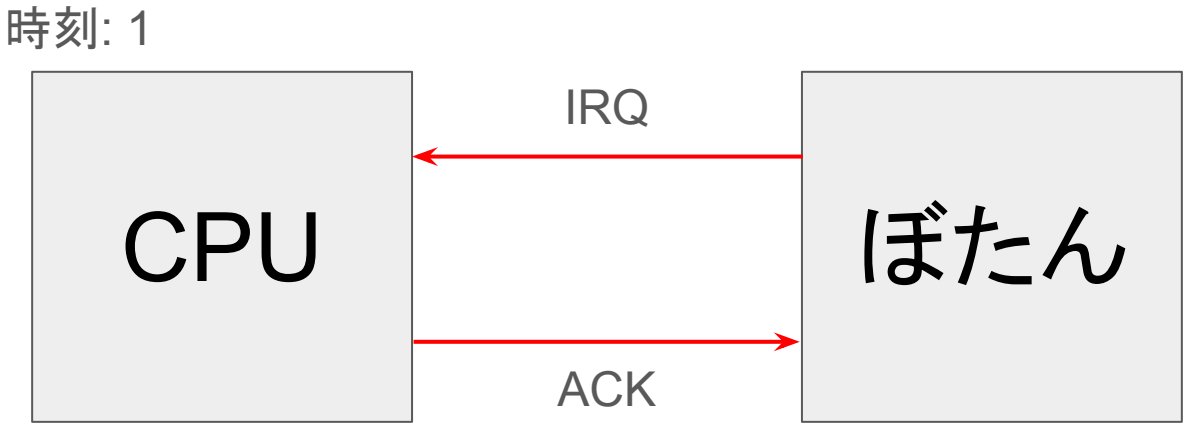
ハードウェア割り込み用ハードウェア

- 単純なのでボタンを使いましょう
- 押したら割り込み要求信号IRQを出します
- 割り込み応答信号ACKが来たら割り込み要求を停止します
- 以上！

CPUとハードウェアの挙動

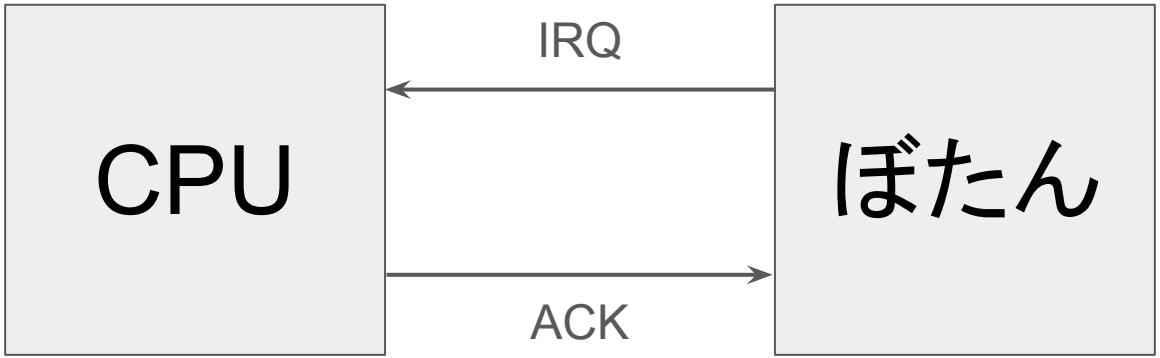


CPUとハードウェアの挙動



CPUとハードウェアの挙動

時刻: 2



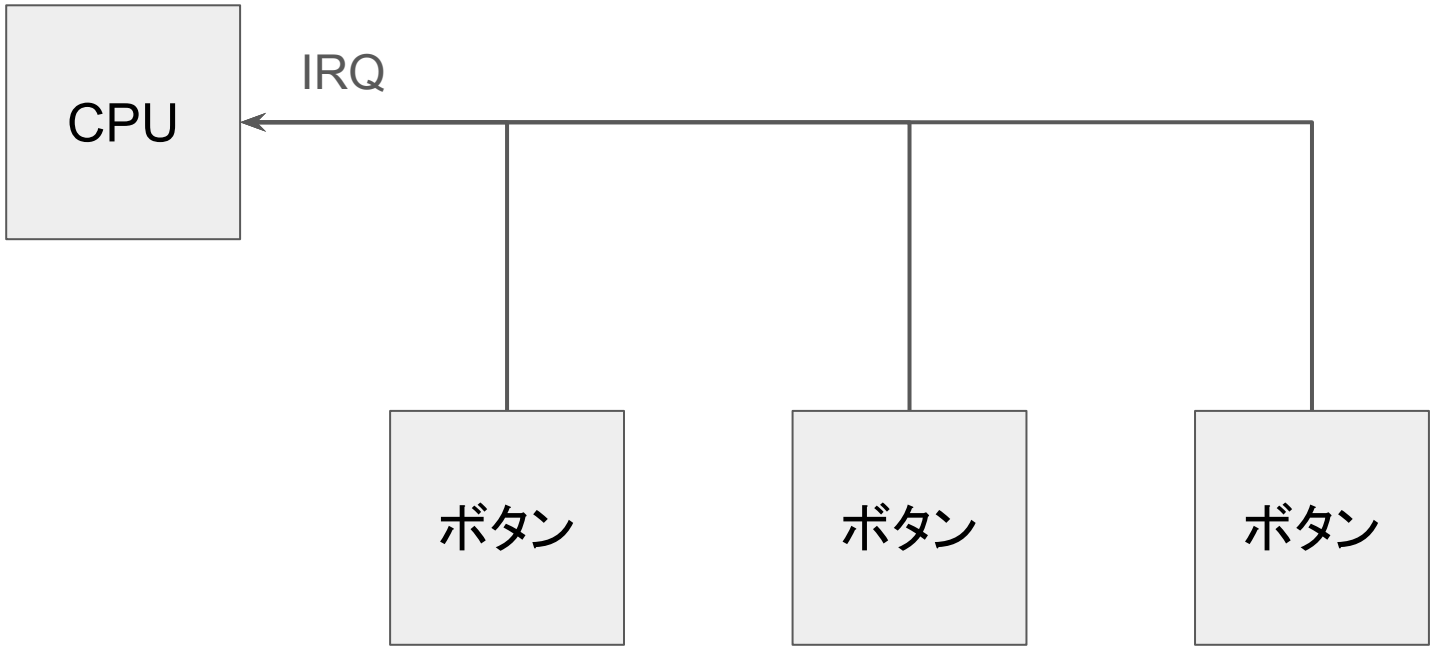
ACKが切れた瞬間IRQを切る！

ハードウェアが複数の場合は？

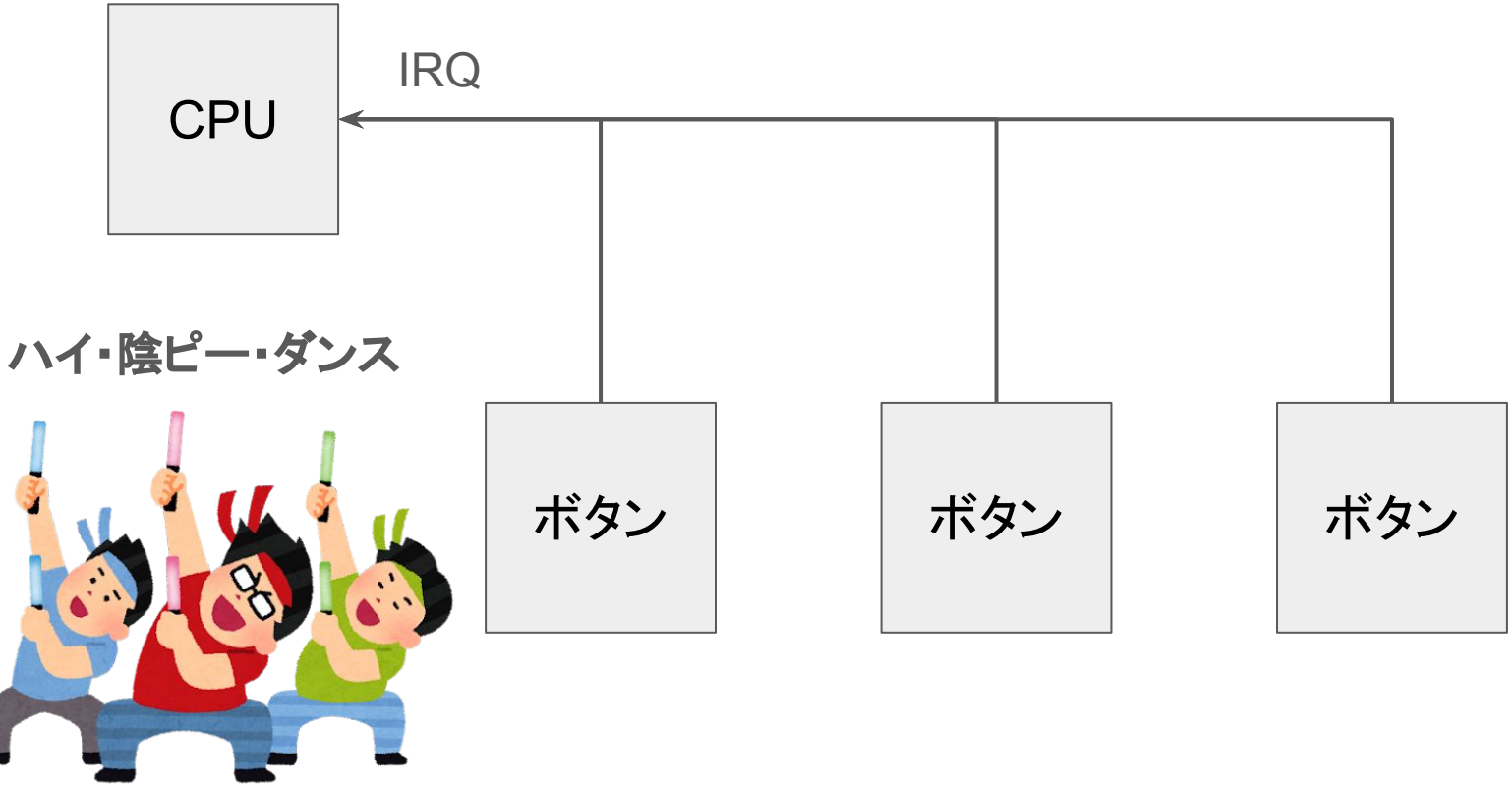
- 同時に割り込みなんてされたら大変
- 割り込み調停を行う必要がある
  - どの割り込みに応答するかを決める
- 今回はレトロゲームにしばしば採用されたデイジーチェーンを紹介

デイジーチェーン

デイジーチェーン

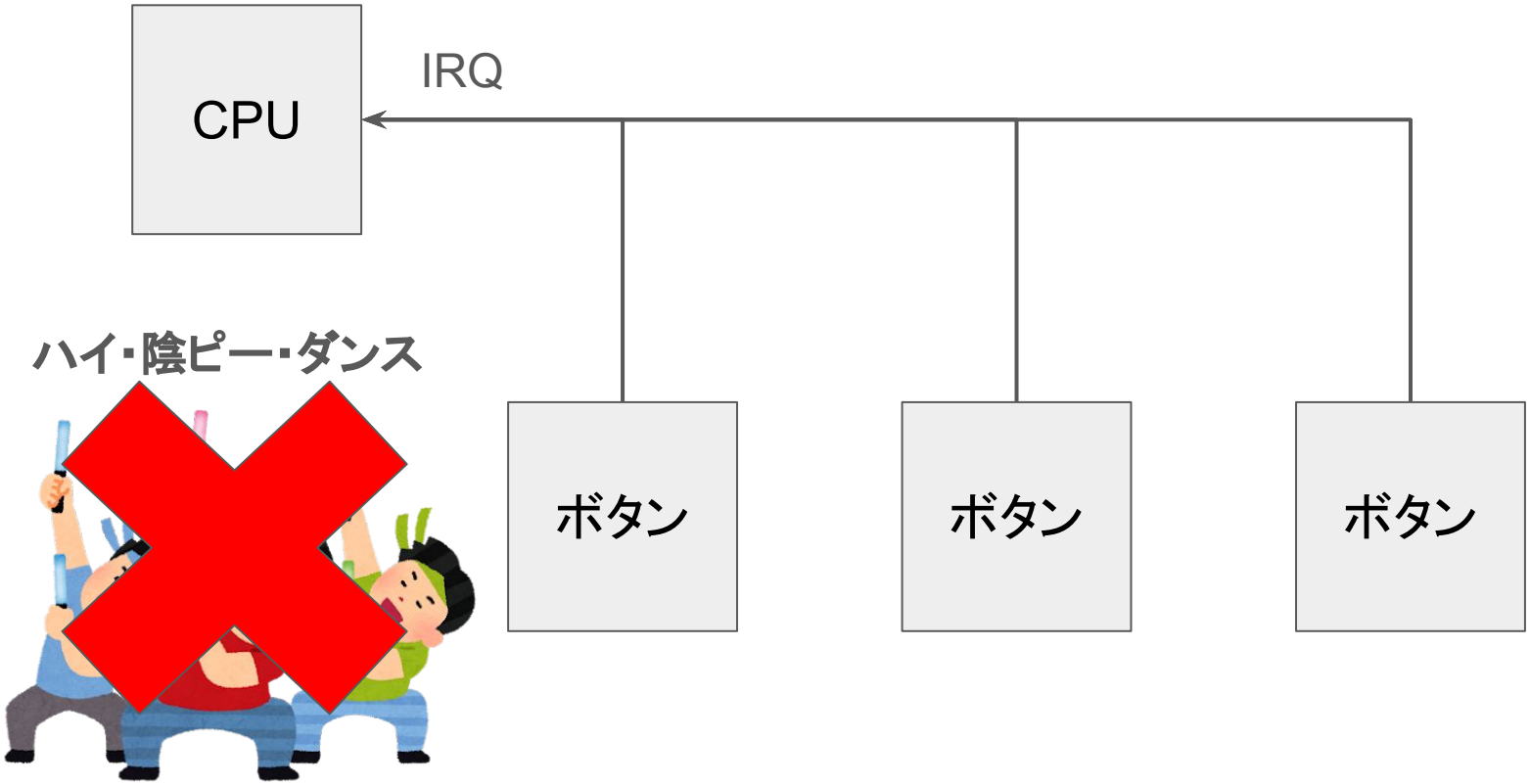


デイジーチェーン

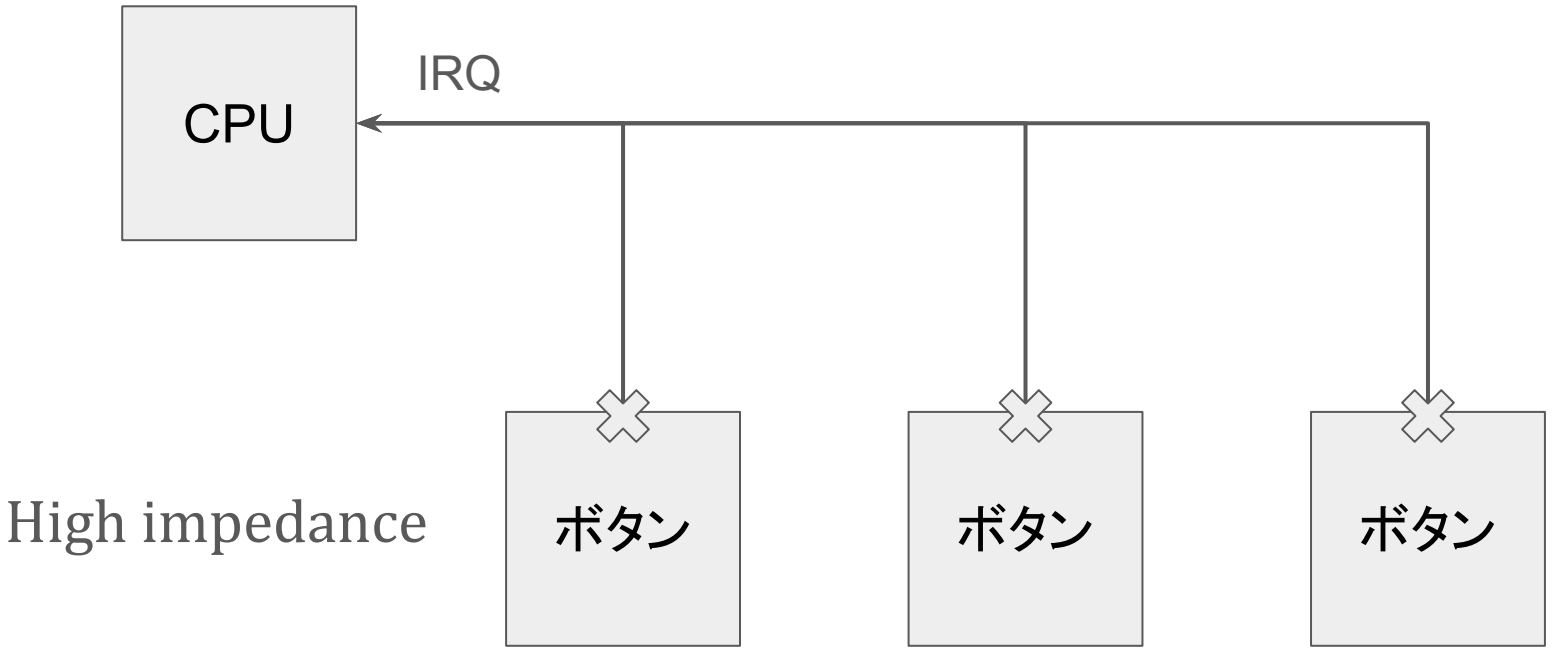


ハイ・陰ピー・ダンス

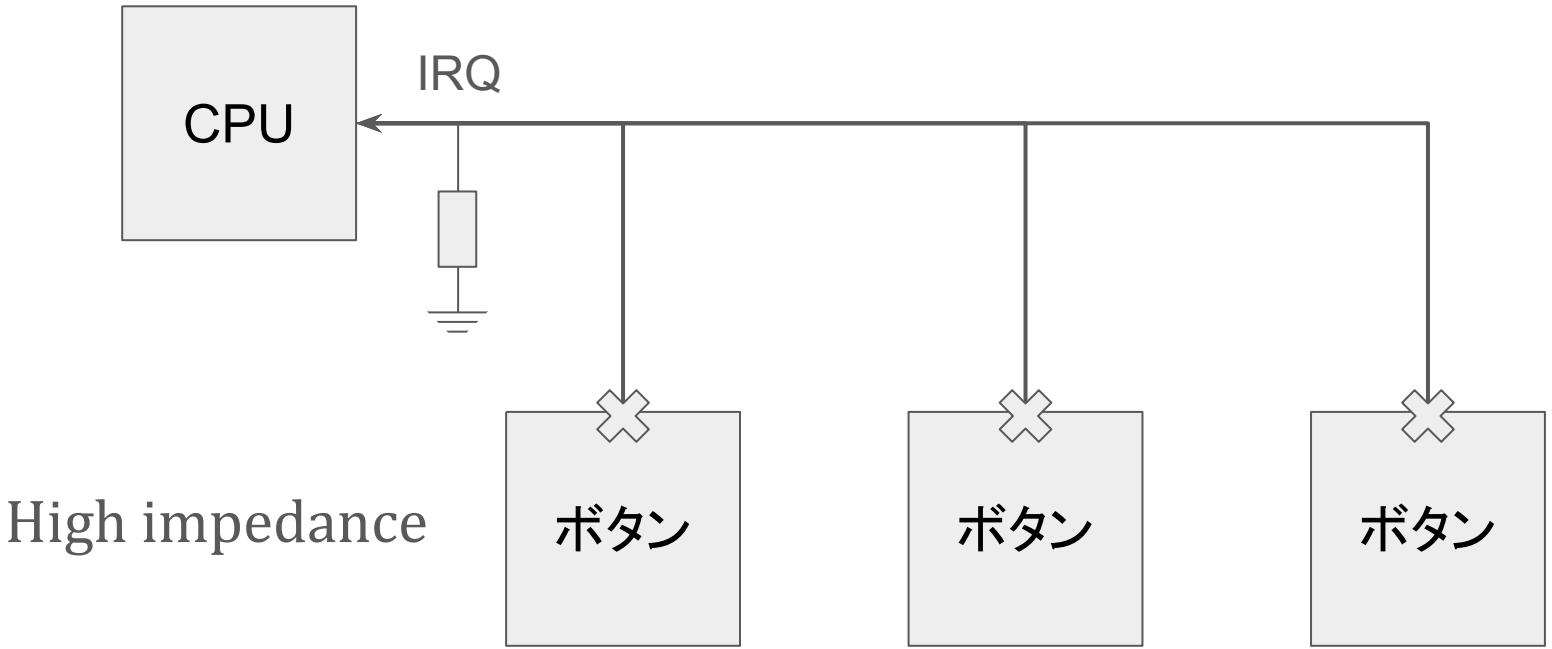
デイジーチェーン



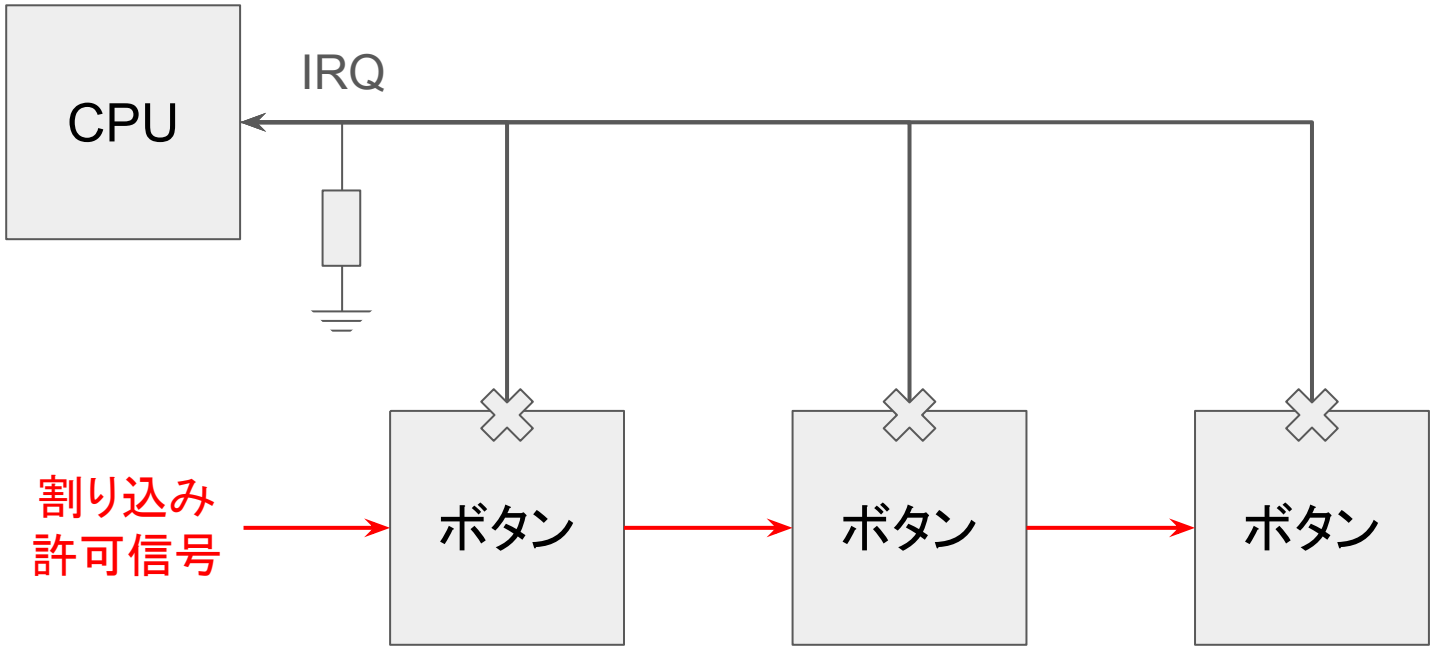
デイジーチェーン



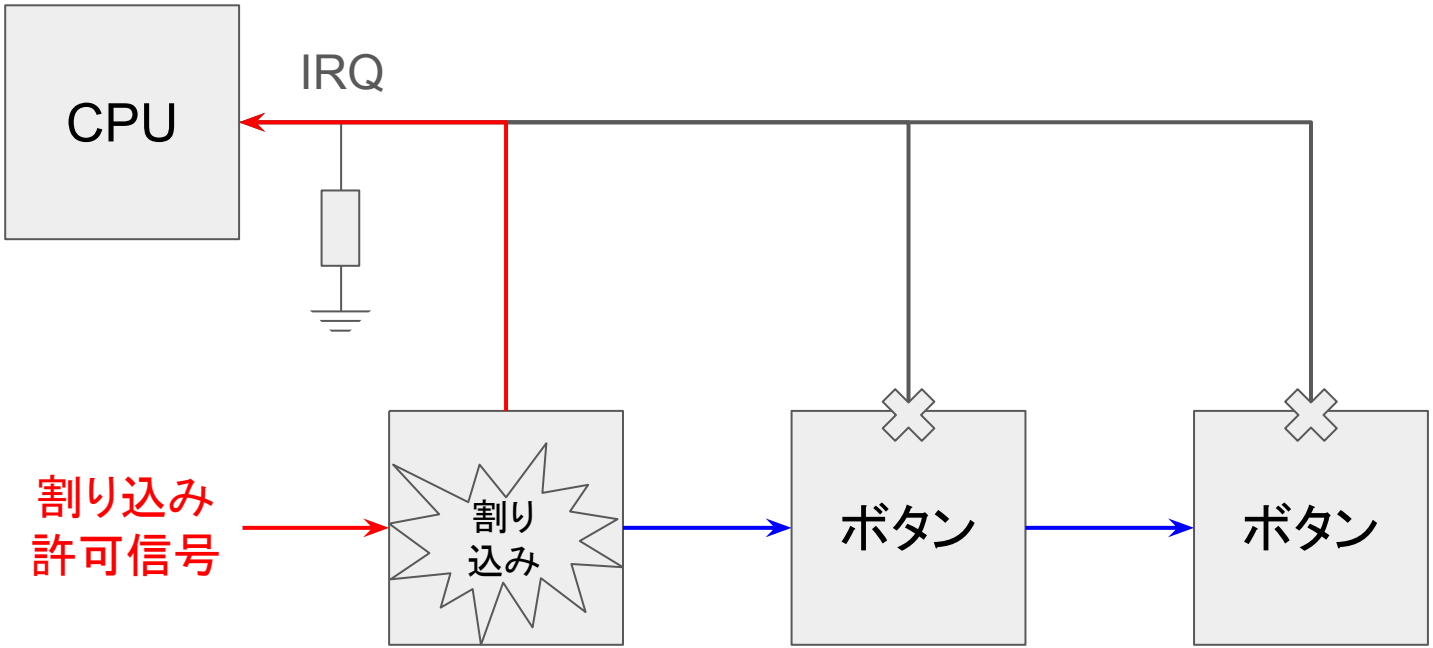
デイジーチェーン



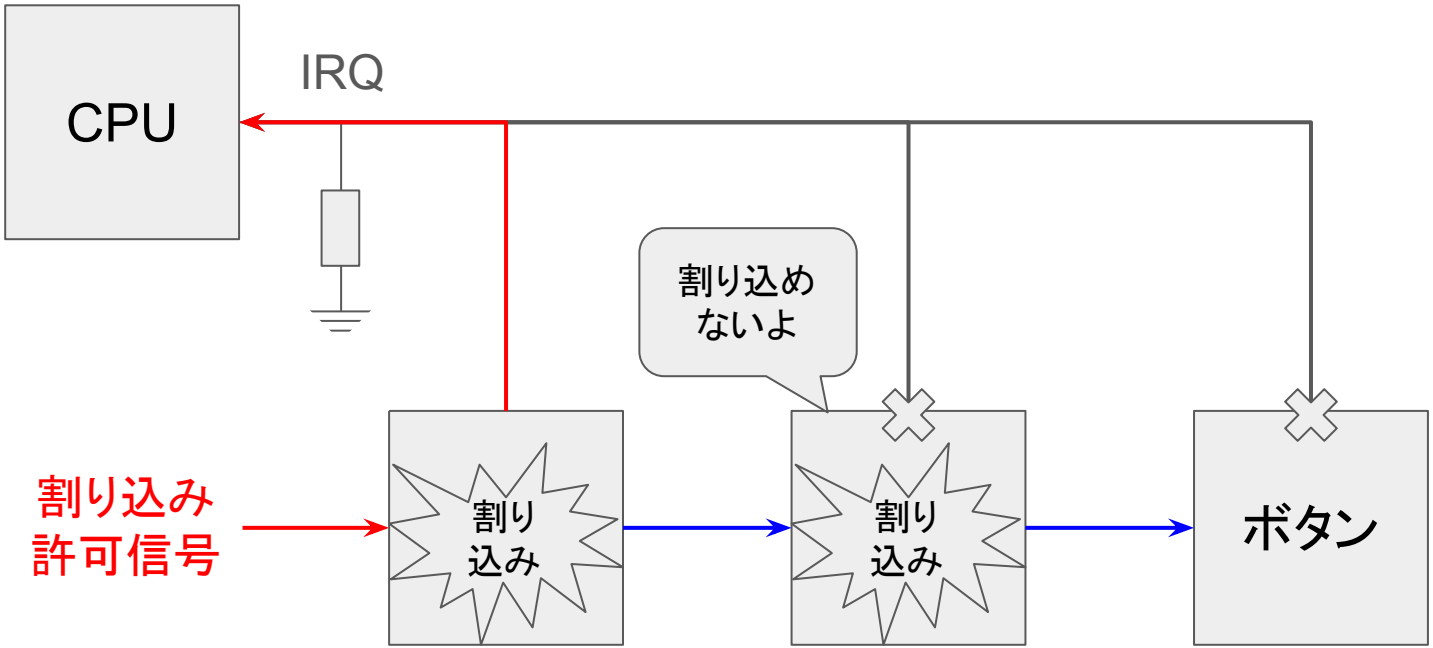
デイジーチェーン



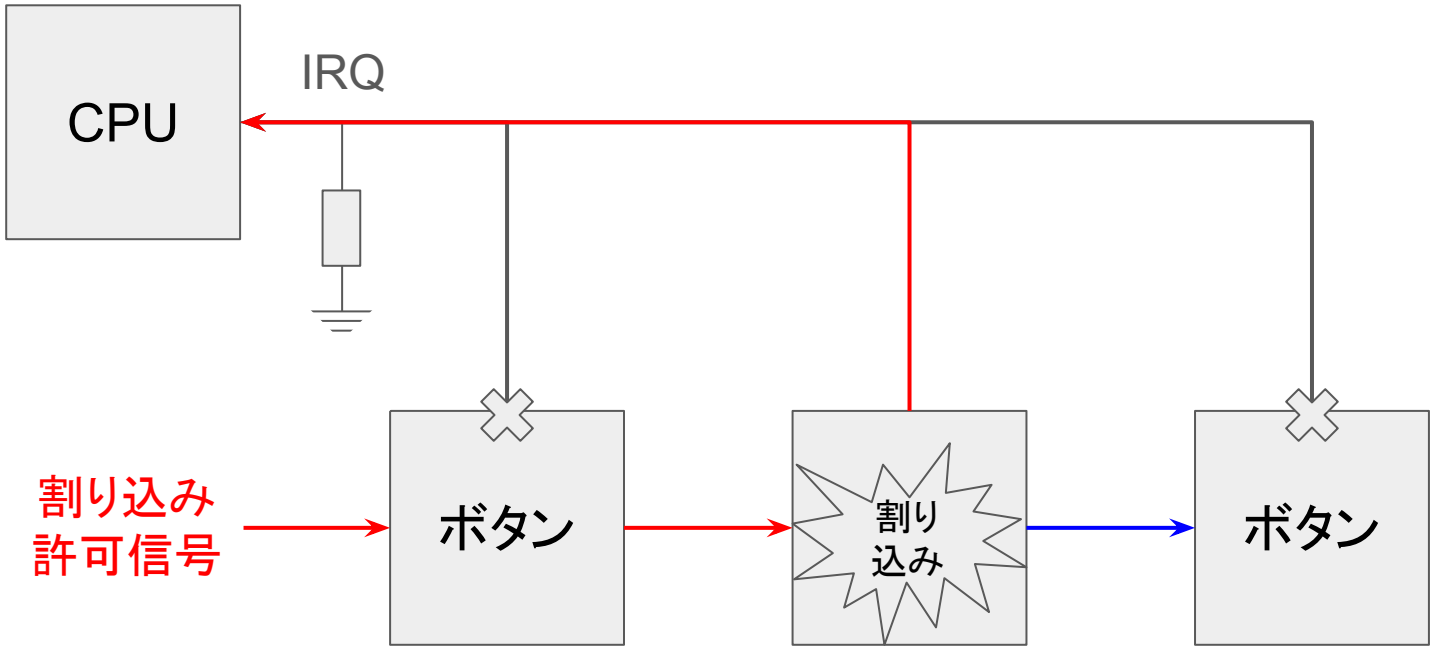
デイジーチェーン



デイジーチェーン



デイジーチェーン



割り込み無効化

- ところで、全ハードウェアの割り込み要求に付き合っていたら日が暮れる
- 欲しい割り込みのみに絞る機能

割り込み無効化

- ところで、全ハードウェアの割り込み要求に付き合っていたら日が暮れる
- 欲しい割り込みのみに絞る機能

割り込み無効化

- 初期状態で全ての割り込みを無効化
- 有効化されたボタンだけ割り込み要求できる
- 割り込み有効/無効を制御できる命令を追加

- ところで、全ハードウェアの割り込み要求に付き合っていたら日が暮れる
- 欲しい割り込みのみに絞る機能

以上がTW4のあらましです

- 今回の話についてこの本に書いてます
- GitHubにソースコードを上げています
  https://github.com/lemoncmd/TW4

ここから買えます→