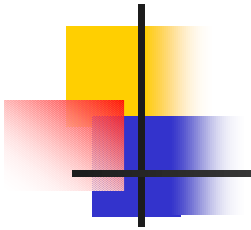
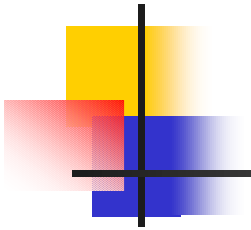
コンピュータの動作入門

和田　知久
琉球大学・工学部・情報工学科教授
wada@ie.u-ryukyu.ac.jp
http://bw-www.ie.u-ryukyu.ac.jp/~wada/



アウトライン

1. ソフトウエアとハードウエア
2. ソフトウエアはどのようにハードウエアで実行されるか
3. コンピュータの基本構成
4. シーケンスのコントロール
5. ノイマン形コンピュータ
6. 一般的RISCコンピュータの構成
7. ミニミップスマシン
8. 実際のPCの構成
9. キャッシュメモリの導入（メモリの階層化）

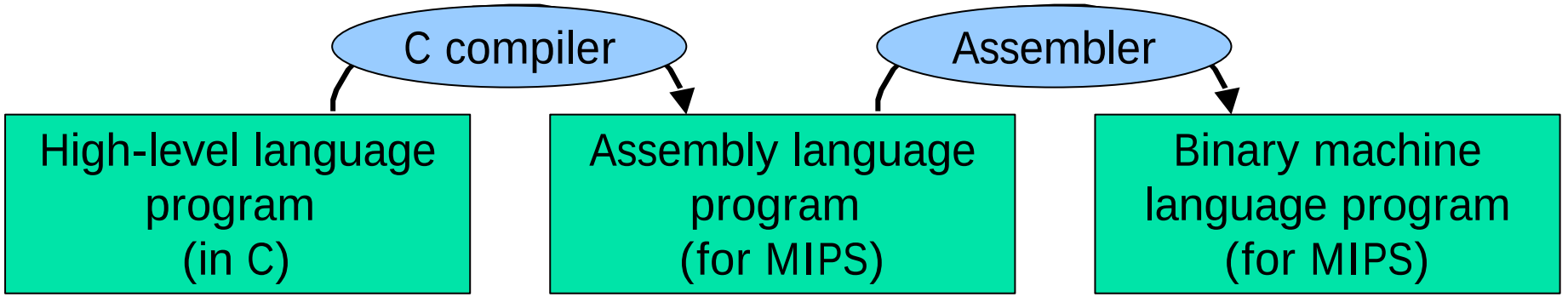


ソフトウエアとハードウエア

- コンピュータ関連には２種類のエンジニアがいる
  ソフトウエア　VS　ハードウエア
- ソフトウエア:
  - C, Java, HTML, PARL, PASCAL…
- すべてのソフトウエアはハードウエア上で実行される
  - PC, DSP board, PDA…
- 普通、ソフトウエア人はハードウエアを理解せず、またその逆も真である
- しかし、本当に凄いコンピュータシステムを実現したいなら、H/WとS/Wを単独に最適化しても無意味である。
- 本当に凄いものをやりたいなら、両方を最適化する必要がある。

さて、S/Wはどのように H/Wで実行される

- デジタルコンピュータは‘1’か‘0’のみ取り扱う




```
Swap (int v[],
int k);
{int tmp;
  temp = v[k];
  v[k] = v[K+1];
  v[k+1] = temp;
}
```

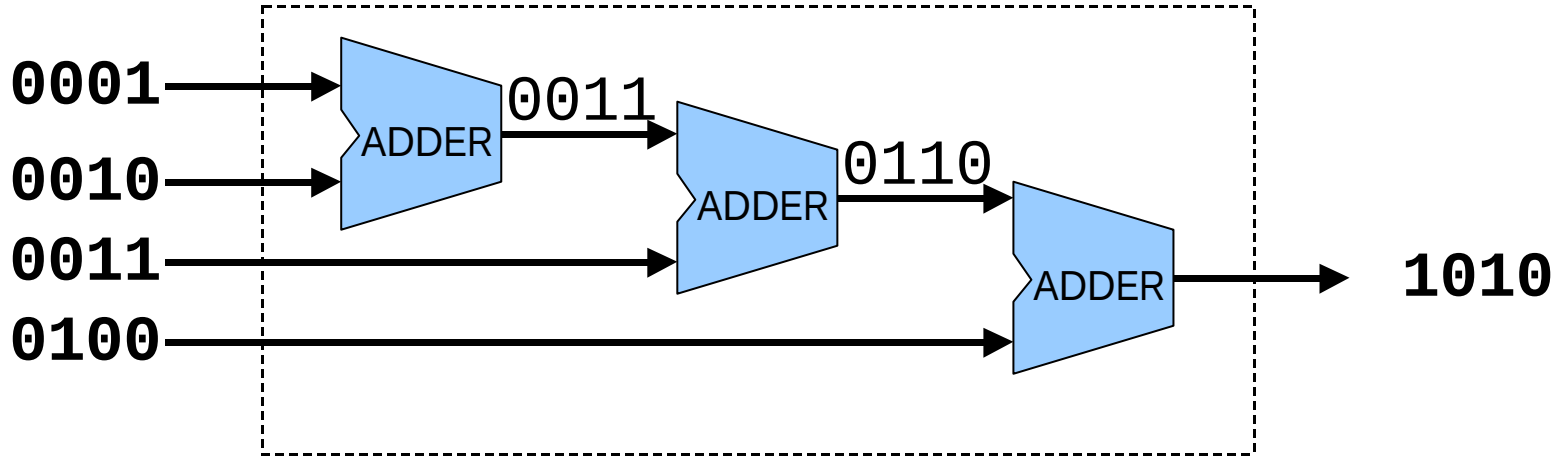
```
Swap:
      muli $2,$5,4
      add  $2,$4,$2
      lw   $15,0($2)
      lw   $16,4($2)
      sw   $16,0($2)
      sw   $15,4($2)
      jr   $31
```

32bit width

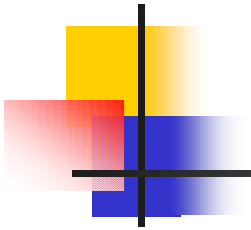
```
00000010100000000000000000000000
00000100011000000000000000000001
10000110001100000000000000000000
10001100111100000000000000000000
10101100111100000000000000000000
10101100011000000000000000000000
00000011111000000000000000000000
```

ケース1：1+2+3+4をはどのように計算できるか？

今、H/W（LSIチップ）があるとすれば、以下のような回路を作れば良い！

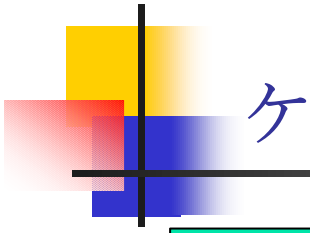


この場合には、ソフトウエアは必要ない！



もし、加算機（ADDER）が１つしかない場合どうする？

PROBLEM #1



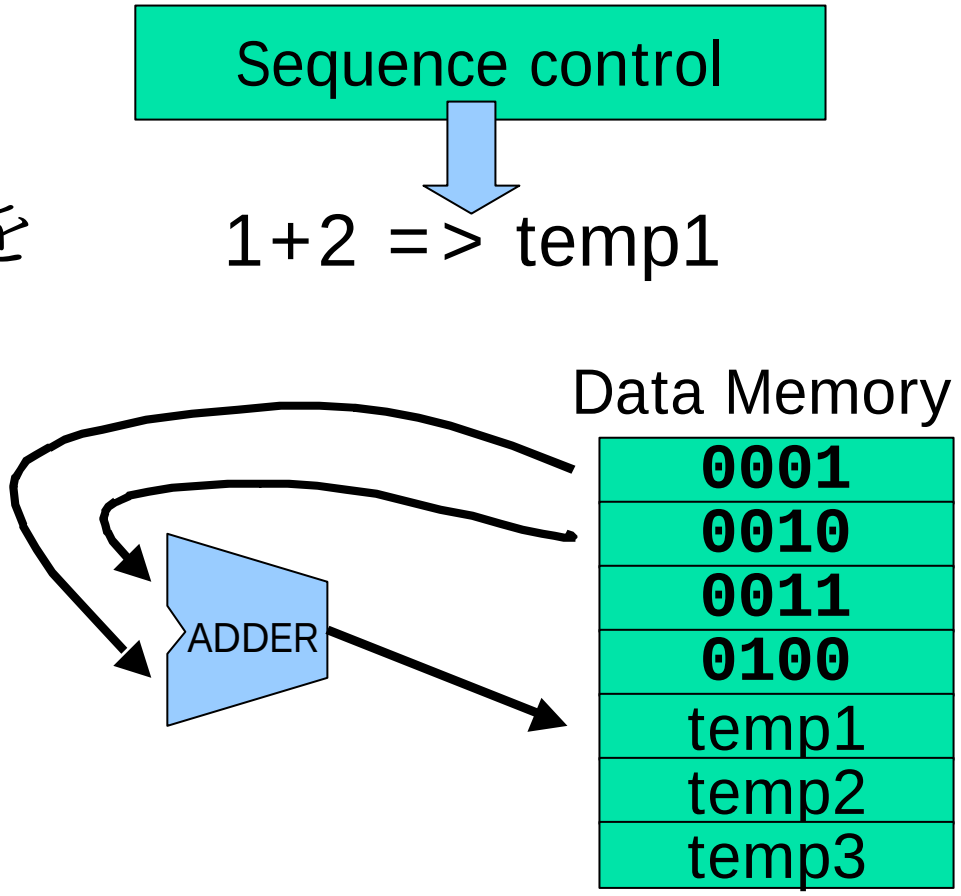
ケース2：1+2+3+4をはどのように計算できるか？

今、加算器が1つしかないとすると…

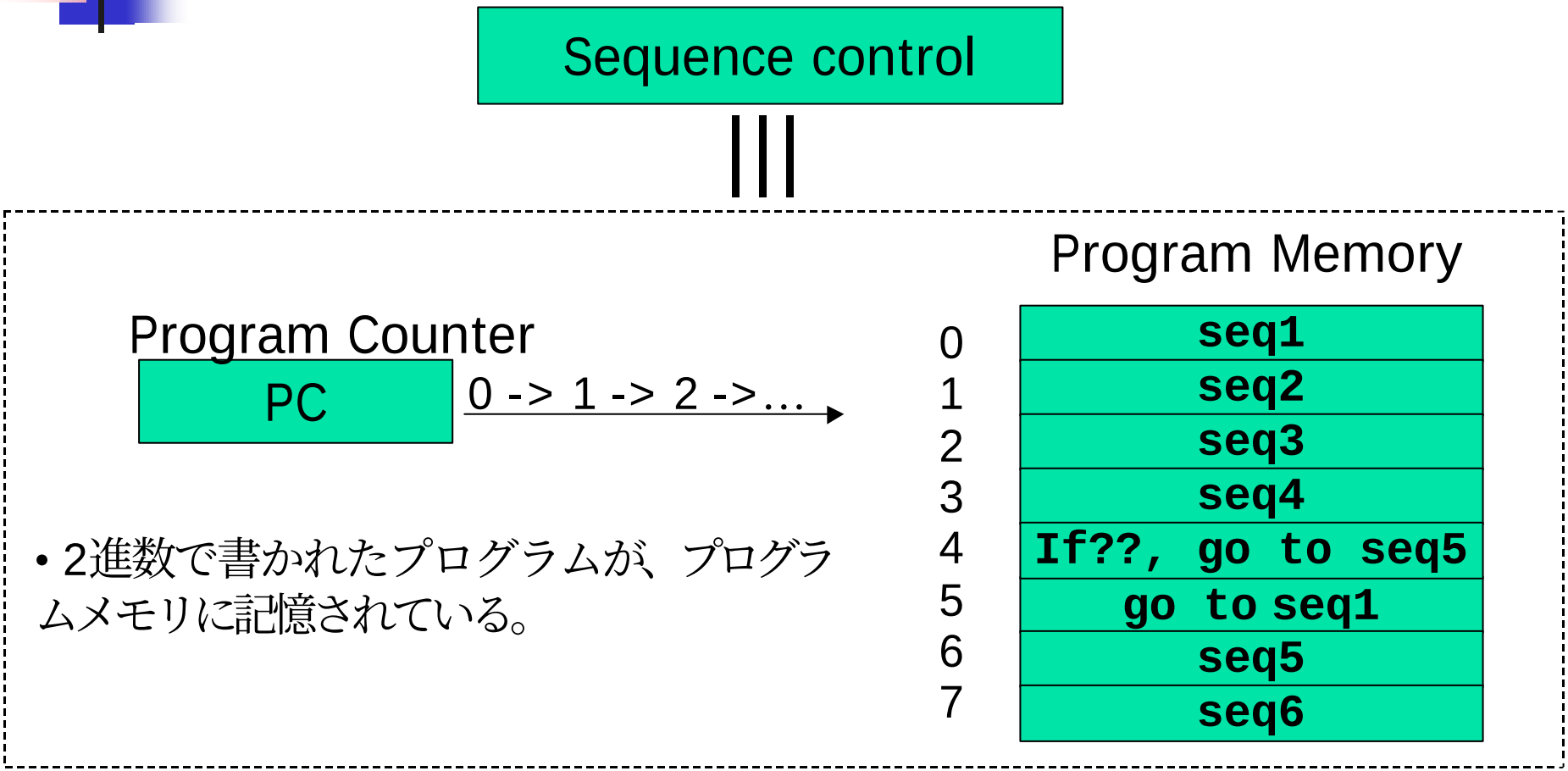
A) 同じ加算器を3回使用する
   1. 1+2 => temp1
   2. temp1 + 3 => temp2
   3. temp2 + 4 => temp3 => 答え
B) 中間の計算値をメモリ領域に記憶する必要がある
C) 上記３つの手順は順序どおりに実行される必要がある

ケース2：1+2+3+4をはどのように計算できるか？

- コンピュータの基本要素（部品）
1. データパス（計算を行うところ）
2. シーケンス（順序）のコントロール
3. 記憶領域（メモリ）



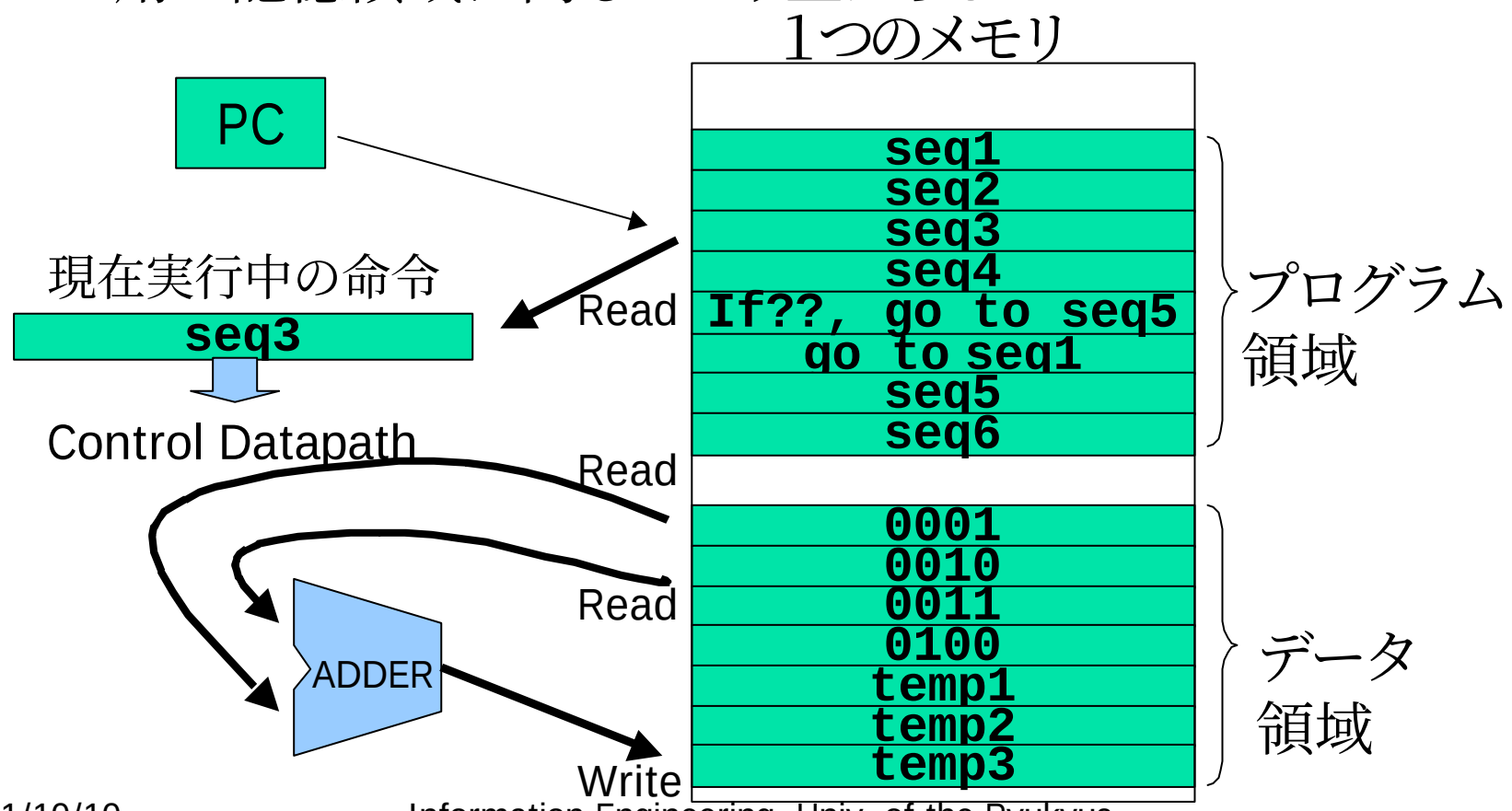
実在コンピュータでの、シーケンスコントロールとは



- 2進数で書かれたプログラムが、プログラムメモリに記憶されている。

プログラムもデータも同じメモリ上にある -ノイマンマシン-

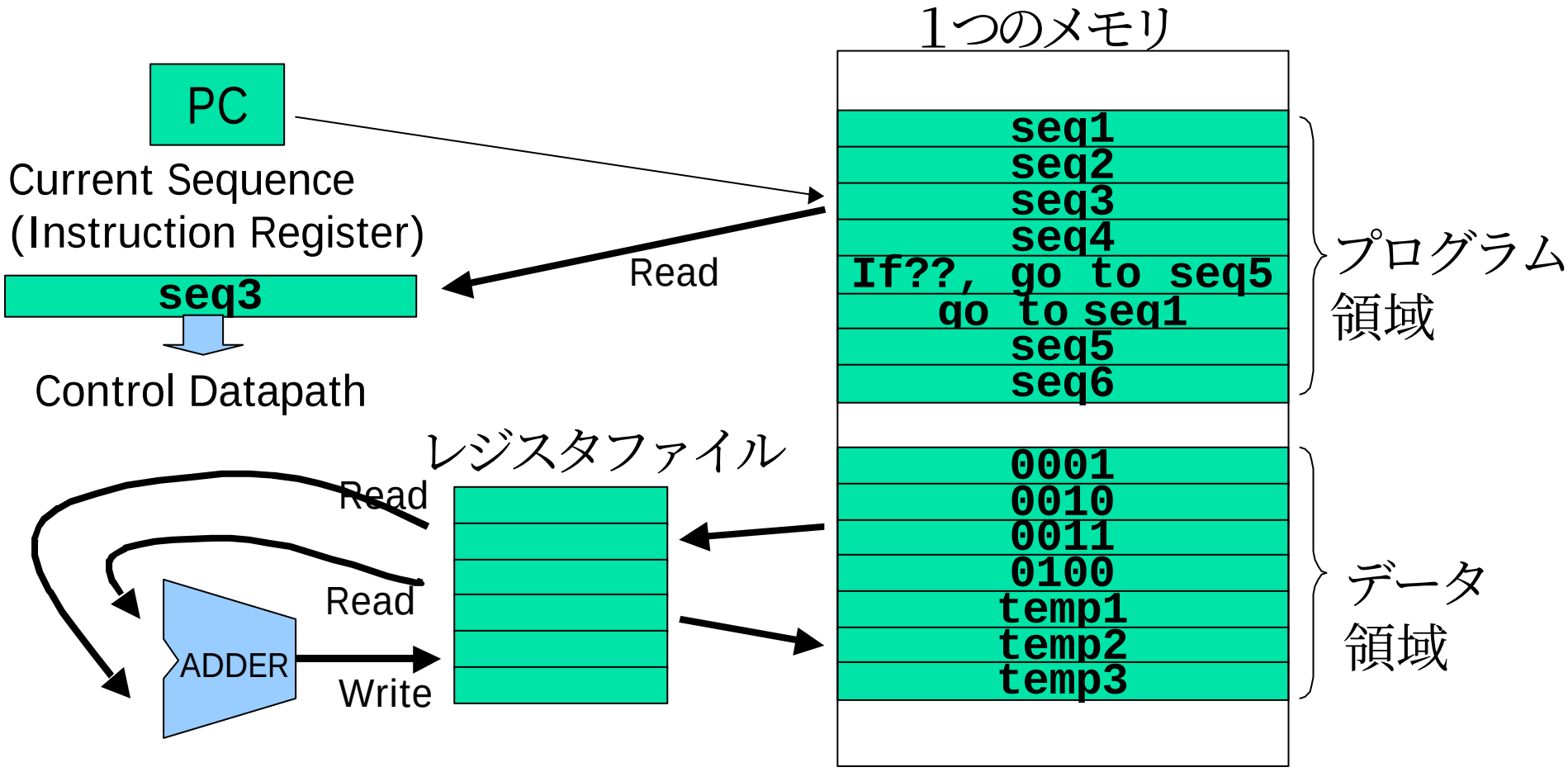
- 現実のコンピュータでは、データ用の記憶領域とプログラム用の記憶領域は同じメモリ上にある

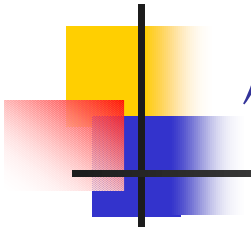


前ページのコンピュータの問題点

- １つの命令を実行する時に,
  命令をメモリから1回読み出す,
  ２つのデータをメモリから読み出す,
  計算結果をメモリに書き込む
- メモリの使用回数が高く、メモリが低速であれば性能が上がらない。
- そこで、データを一時的に記憶するレジスタファイルを設けた
  - メモリ: たとえば１２８メガバイト
  - レジスタファイル: たとえば１２８バイト
- それにより、新しい命令が必要となった
  - Load Word (LW)：データ領域からレジスタファイルにデータ転送
  - Store Word (SW)：レジスタファイルからデータ領域にデータ転送

一般的なRISC型コンピュータの構成



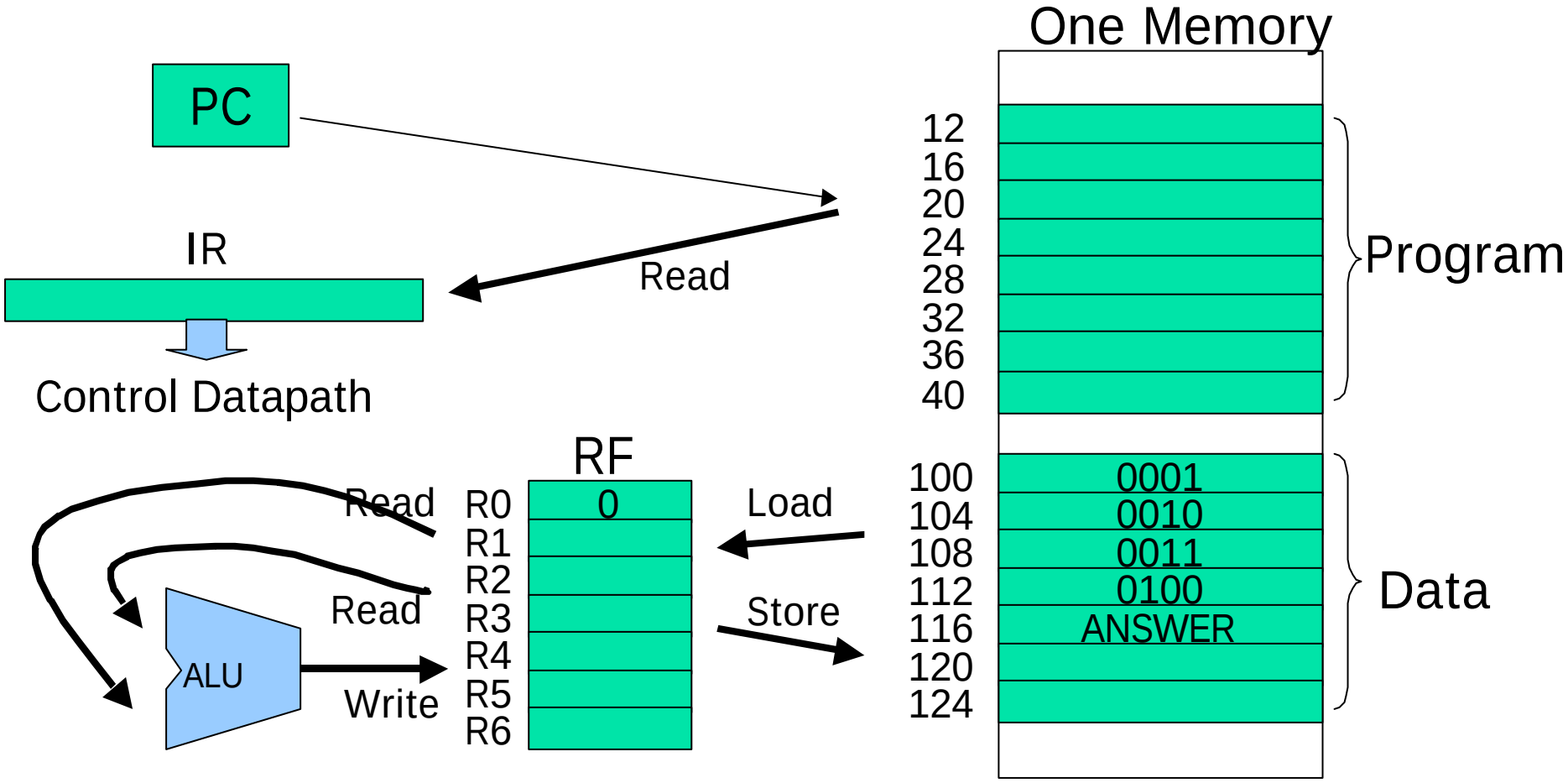


例: ミニ　ミップス マシン

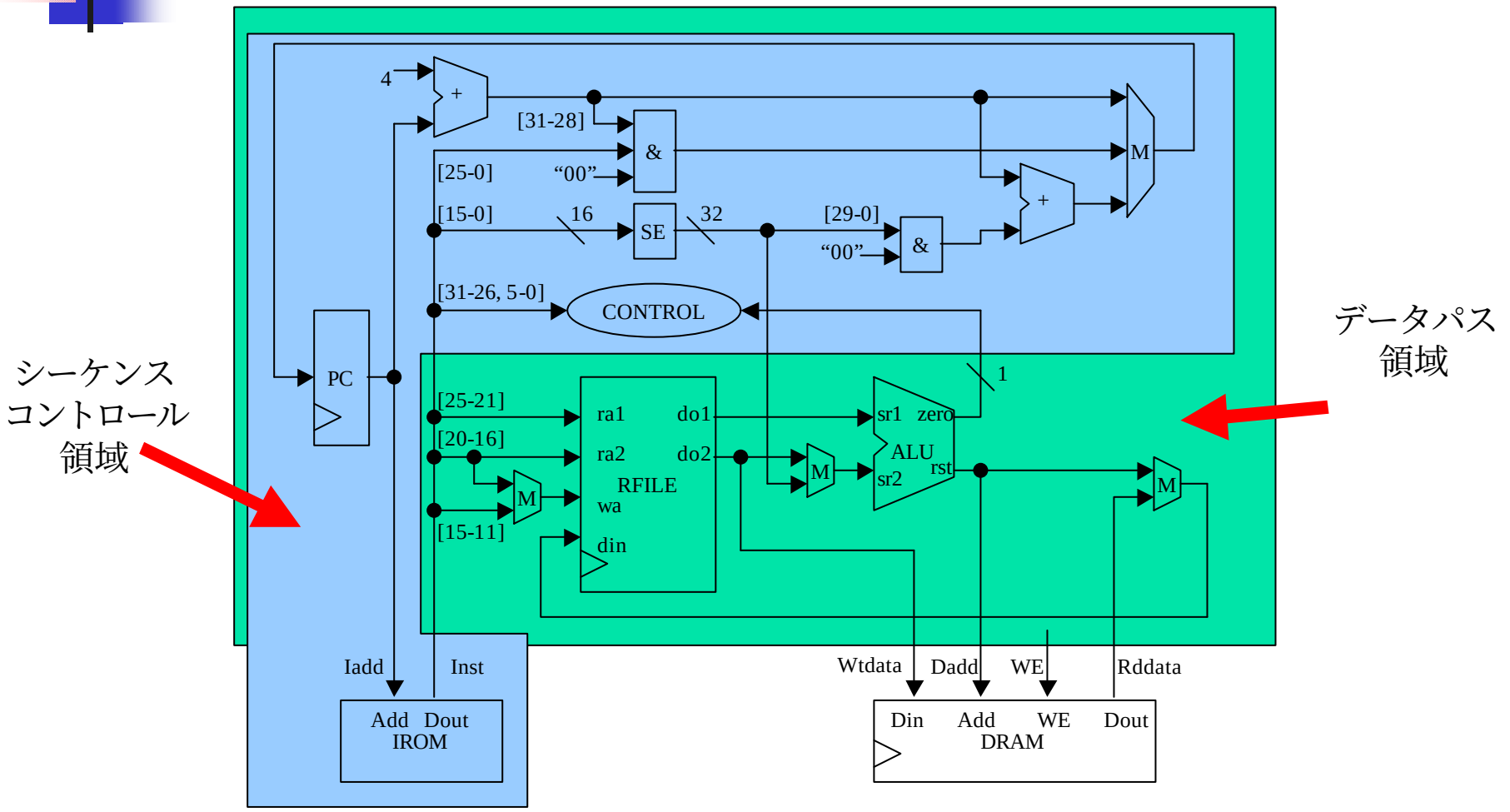
32個のレジスタR0,R1,…R31を持つRISCコンピュータ

Category	Instruction	Example	Meaning	Comments
算術命令	add	add R1,R2,R3	R1 <= R2 + R3	3 operands
	subtract	sub R1,R2,R3	R1 <= R2 - R3	3 operands
論理命令	and	and R1,R2,R3	R1 <= R2 and R3	3 operands, bit-wise logical and
	or	or R1,R2,R3	R1 <= R2 or R3	3 operands, bit-wise logical or
データ転送	load word	lw R1, 100(R2)	R1 <= Memory[R2+100]	Data from memory to register
	store word	sw R1, 100(R2)	Memory[R2+100] <= R1	Data from register to Memory
条件分岐	branch on equal	beq R1,R2,25	if (R1=R2) go to PC+4+100	Equal Test; PC relative branch
	set on less than	slt R1,R2,R3	if (R2<R3) R1<=1 else R1<=0	
無条件分岐	jump	j 2500	go to 10000	Jump to target address

ミニ　ミップス命令をもちいて1＋2＋3＋4を計算するプログラムを作る！

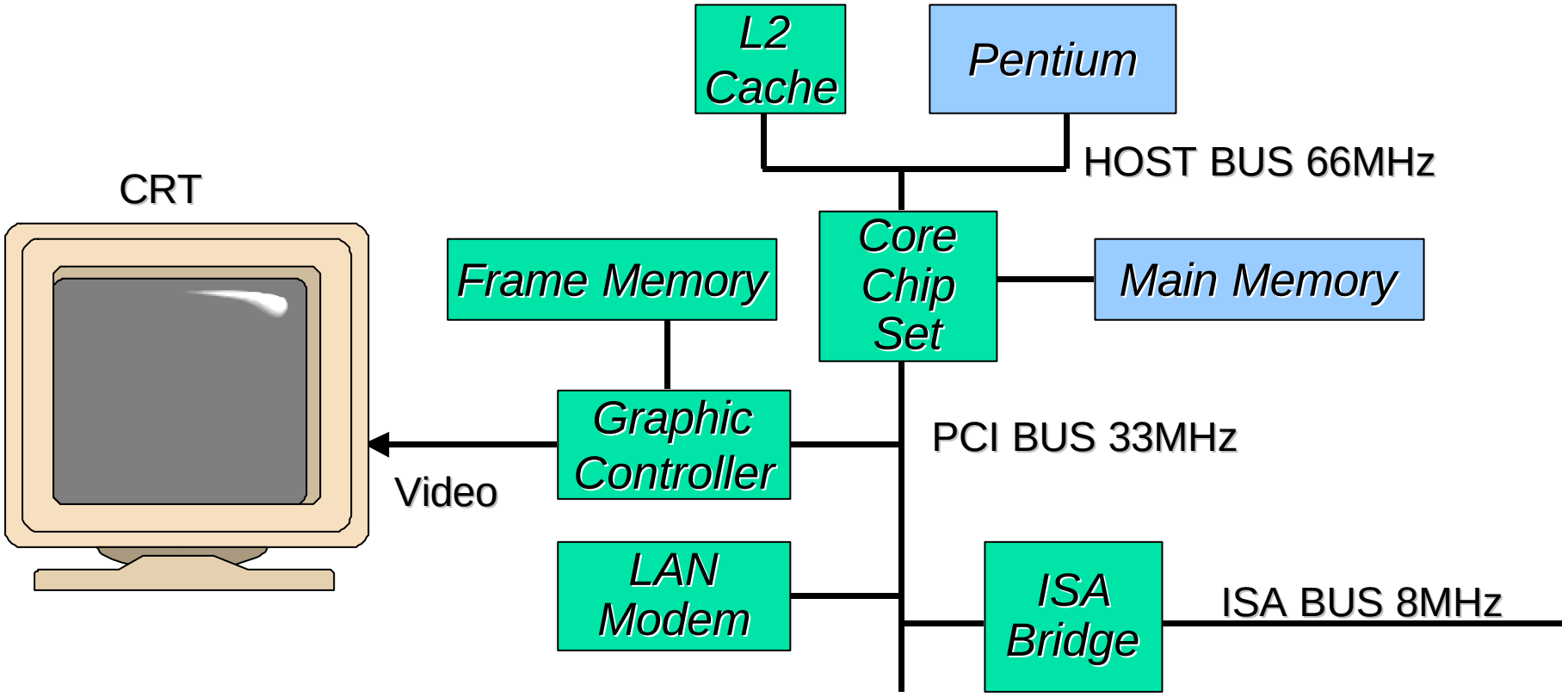


現実のミニミップスマシンのブロック図



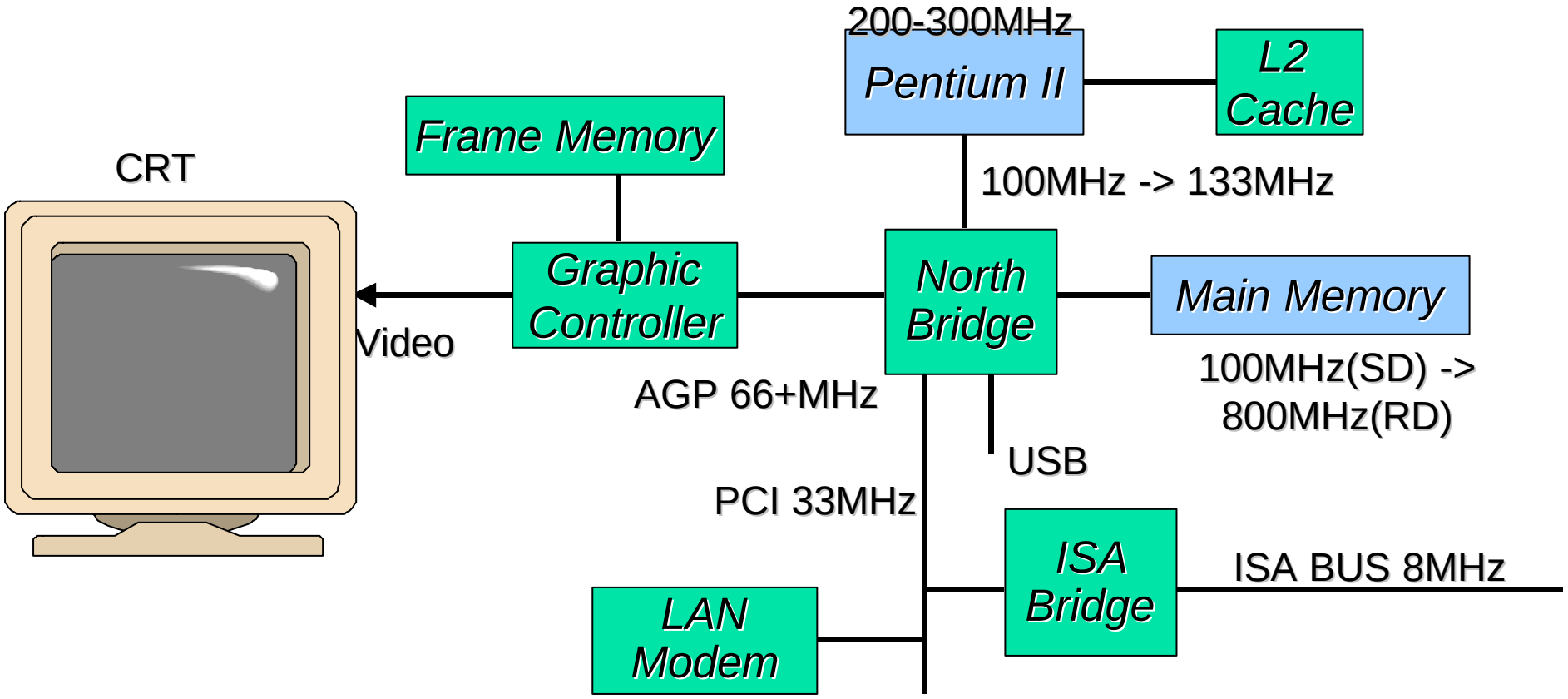
現実の Pentium PC の構成 @ 1995

- 各四角形はLSIを示す
- 線はバスと呼ばれ、多数の配線の束である。

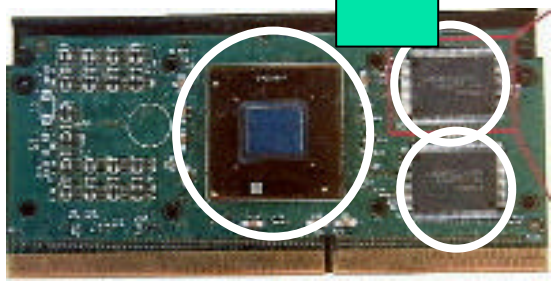


現実のPentiumII PCの構成 @ 1997

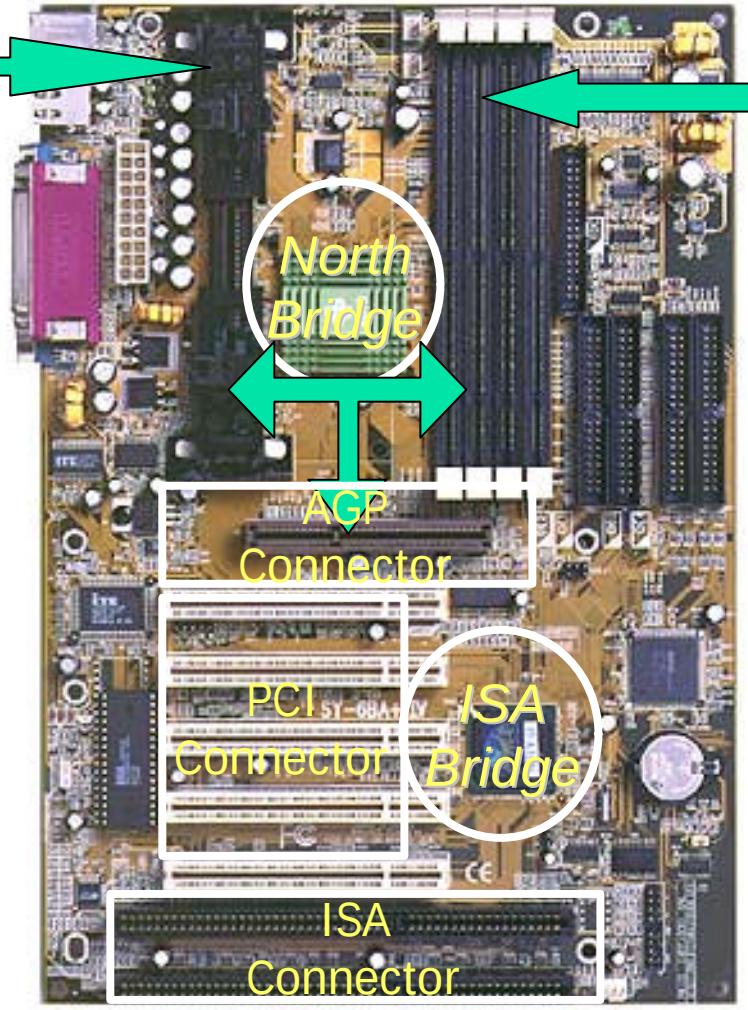
- 各四角形はLSIを示す
- 線はバスと呼ばれ、多数の配線の束である。.



Real PentiumII Mother board

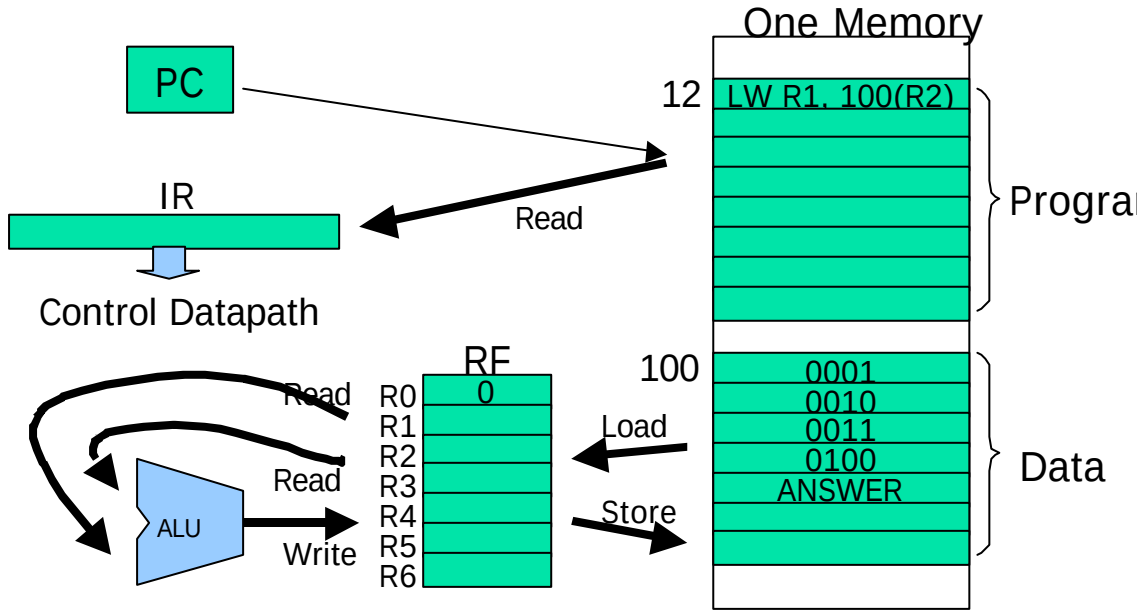


Pentium II and
Cache module



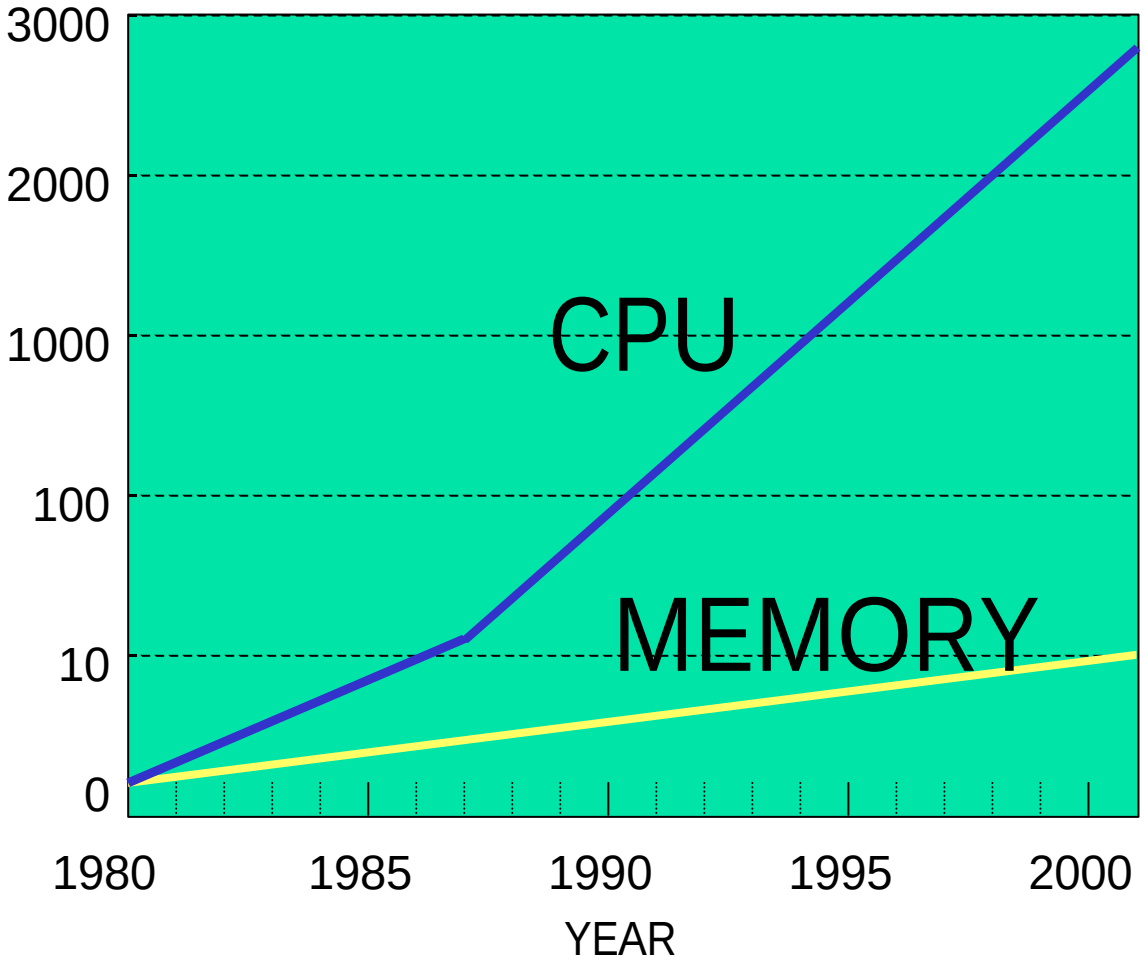
もう一度動作の確認！

LW R1, 100(R2) を実行すると実際にはどう動くか？

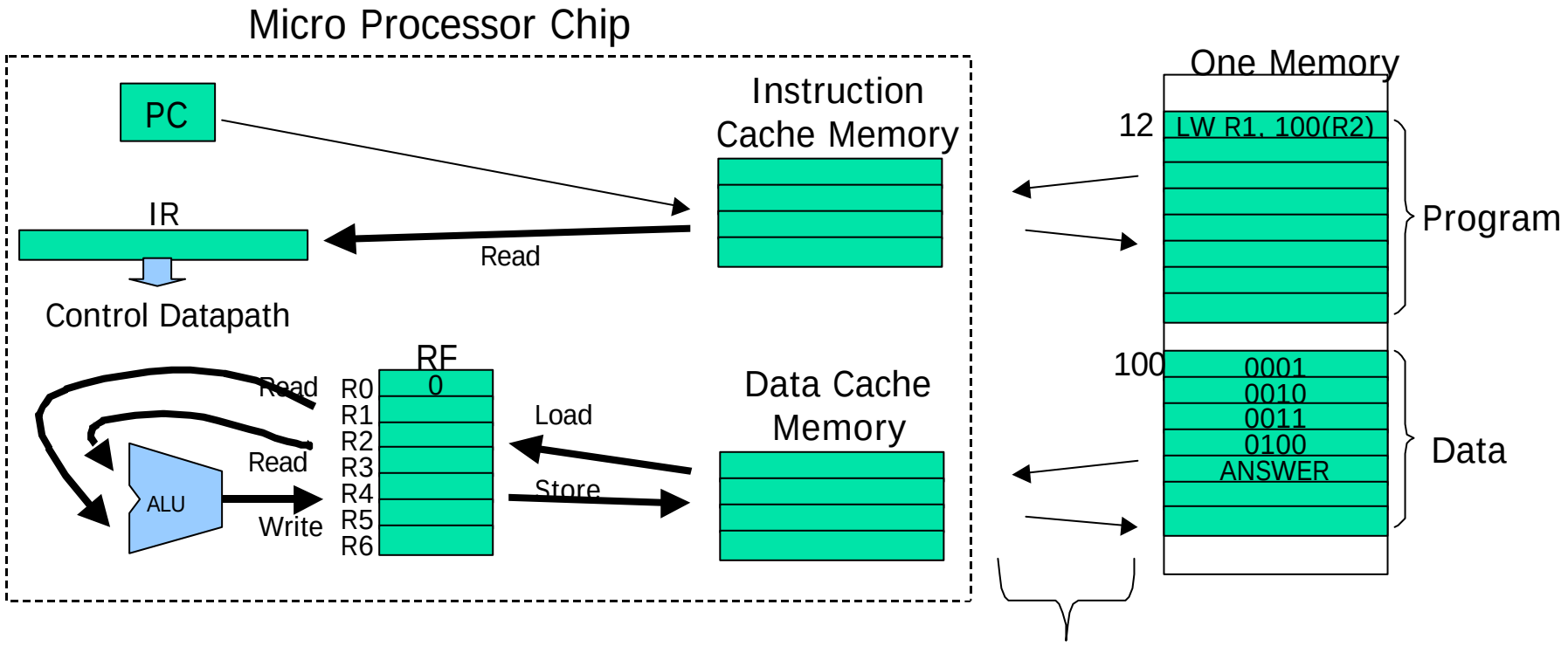


- PC <= 12
- Read memory[12]
- IR <= LW R1, 100(R2)
- ALU compute 100+R2
- Read memory[100+R2]
- R1 <= memory[100+R2]

スピード向上の歴史: メモリ vs. CPU

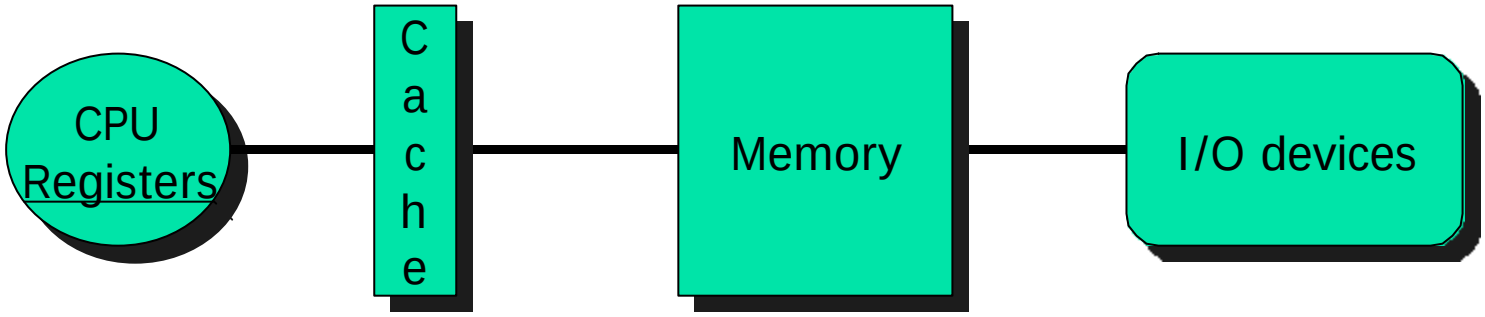


キャッシュメモリが導入
- ハーバードアーキテクチャ -



メモリ（主記憶）とキャッシュメモリ間のデータの転送は
S/Wでプログラムせず、H/Wでコントロールする。

メモリの階層構成で全体性能をUP



	Register reference	Cache reference	Memory reference	Disk Memory reference
Size:	400Byt	256KByte	128MByt	10GByte
Speed:	2ns	5ns	100ns	5ms