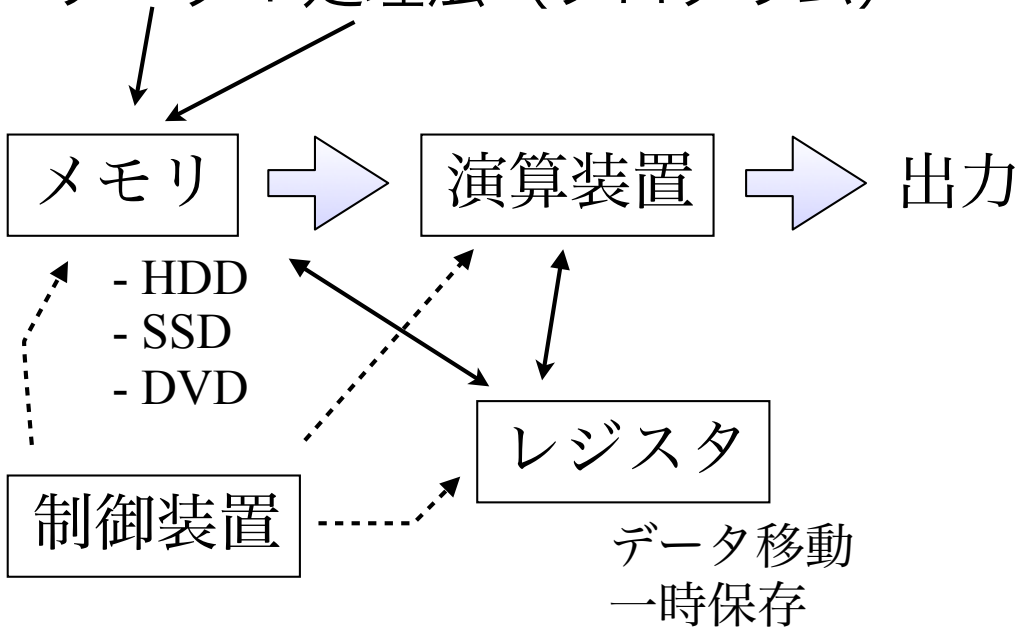
モデルコンピュータ

- ソフトウェア＝データ＋処理法（プログラム）

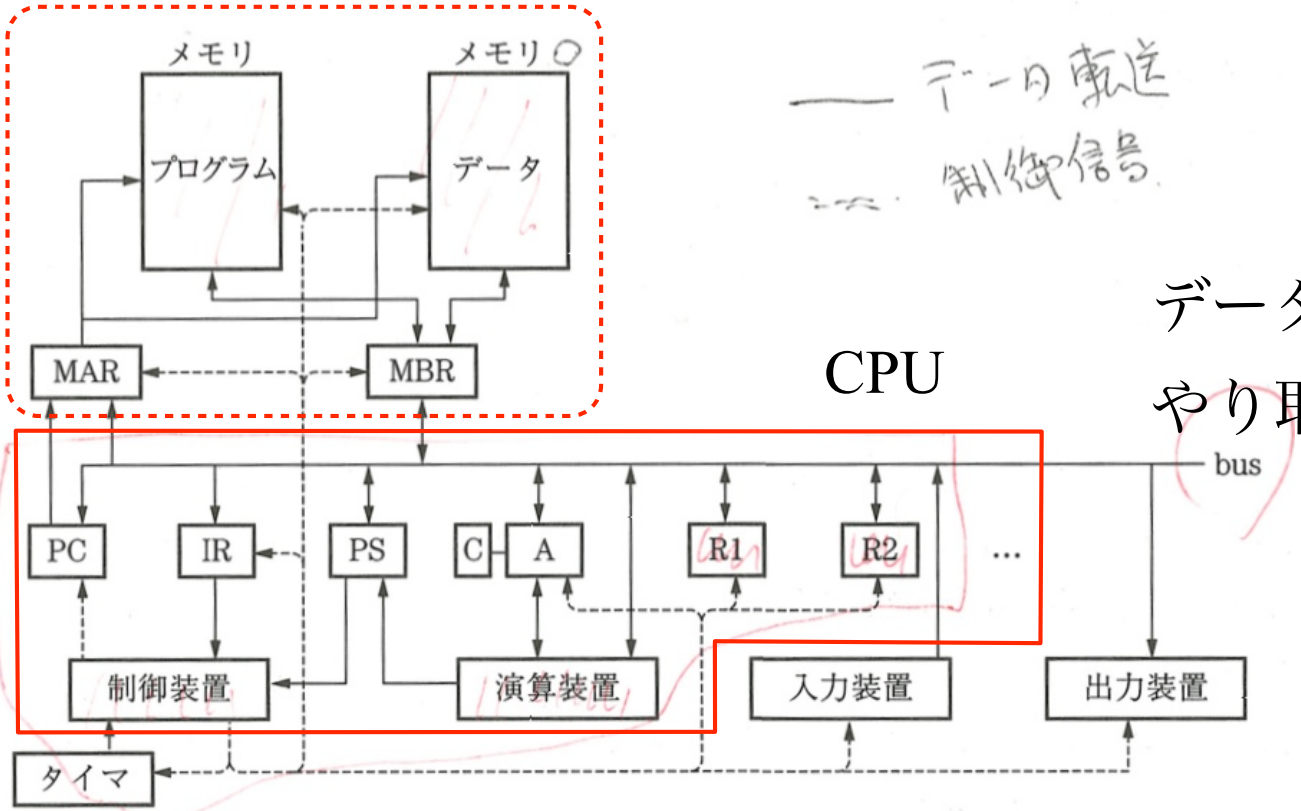


この４つの要素を適切につないだもの＝コンピュータ
つなぎ方や装置の構造構成（**アーキテクチャ**）にはいろいろある

ノイマン型コンピュータ
プログラム内蔵型（データとプログラムの区別無し）
逐次処理

コンピュータの構成例

メモリは次ページ



データがbusを介してやり取りされる

MAR: Memory Address Register　MBR: Memory Buffer Register

PC: Program Counter　IR: Instruction Register

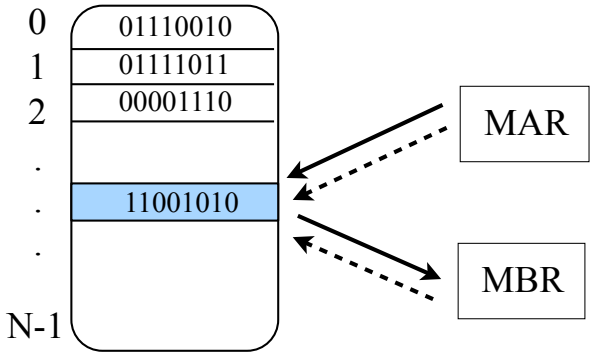
PS: Program Status Register　Ri: Register i (i=0,1,2,···)(R0 = A) レジスタ

C: Carry register　A: Accumulator

図 5.1　コンピュータの構成

メモリの構成とその読み書き

アドレス:メモリ上の場所



N: メモリのサイズ（記憶容量）

readの時

(1) MARにアドレスを置く
(2) read信号がでる
(3)アドレスで指定された領域にある0-1系列をMBRに転送

writeの時

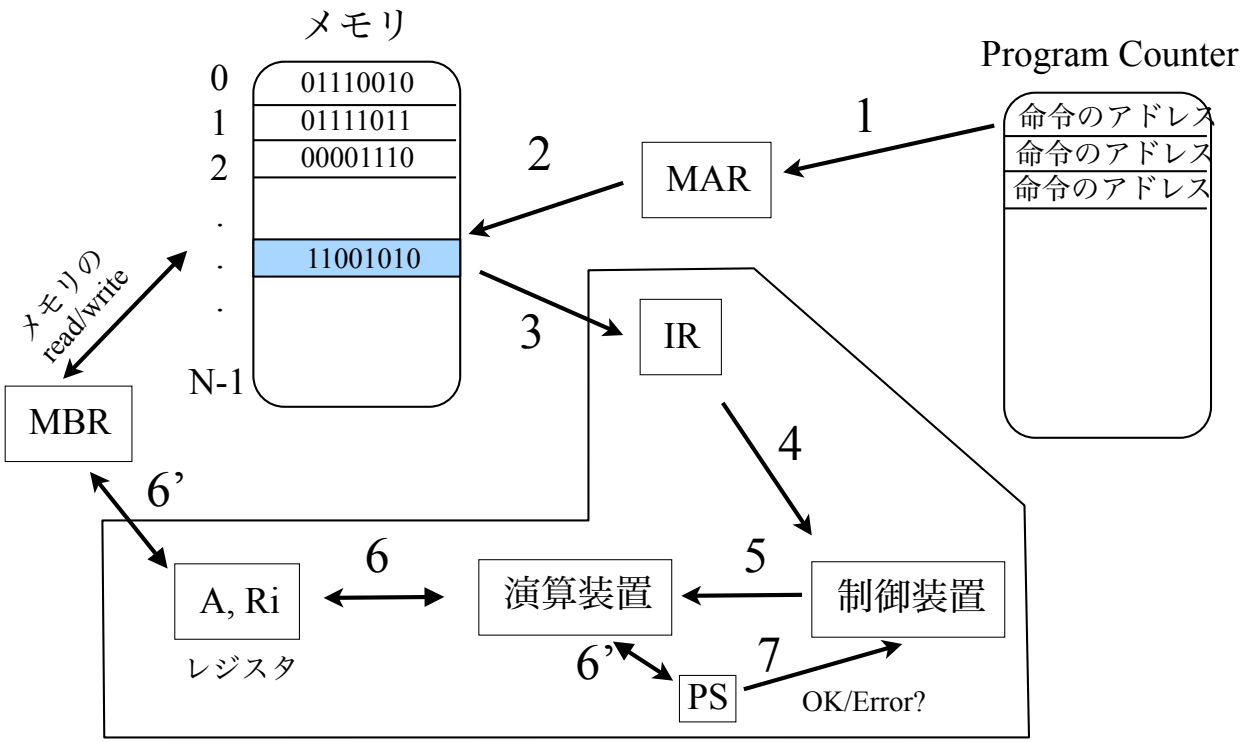
(1) MARにアドレスを置く
(2) write信号がでる
(3)アドレスで指定された領域にMBRから0-1系列を転送

MAR: memory address register
MBR: memory buffer register

- データやプログラムの区別無く、MARにあるアドレスで指定された領域に存在する0-1系列をMBRとやり取りする

ノイマン型コンピュータ
プログラム内蔵型（データとプログラムの区別無し）
逐次処理

命令実行の流れ



IR: Instruction Register
PS: Program Status Register

busを介したデータ転送

- ノイマン型ではプログラムもメモリに入ってる
- プログラムの実行は、PCに入っている命令のアドレスに従って、逐次メモリにある命令を実行していく

機械語命令

- プログラム（機械語命令）も当然のごとく0-1の並び
- **ニーモニックコード：**読みにくいので機械語命令に名前を付けたもの
- **アセンブラ**：ニーモニックコードを機械語命令に変換するプログラム
- 書き方（文法）までひっくるめて、**アセンブラ言語**と呼ばれる
  - 文法と言っても、上から順に並べるだけ
- ニーモニックコードと機械語命令（0-1系列）の対応はCPU毎に異なるので、ここでは扱わない

ニーモニックコード

命令語の形： ラベル ： 命令種別　オペランド (変数名 x, ラベル α, 数字列 n)

データ処理命令

命令語	意味	命令語	意味	
LOAD x	A ← M[x]	SRA n	A ← A >>$_a$ n	Shift R Arithmetically
STO x	M[x] ← A	SRL n	A ← A >> n	Shift R Logically
ADD x	A ← A + M[x]	SLL n	A ← A << n	Shift L Logically
SUB x	A ← A − M[x]	RLL n	A ← A <<$_c$ n	Rotate L Logically
READ x	M[x] ←キーボード	RRC n	A ← CA >>$_c$ n	Rotate R with Carry
WRITE x	プリンタ← M[x]	RLC n	A ← CA <<$_c$ n	Rotate L with Carry
INC	A ← A+1			(R: right (>>))
DEC	A ← A−1			(L: left (<<))
LDI n	A ← n			

プログラム制御命令

命令語	意味	
JMP α	PC ←α	Jump (or Branch)
JZERO α	if A = 0 then PC ←α else next	Jump if Zero
JGTZ α	if A >0 then PC ←α else next	Jump if Greater Than Zero
BSS α	M[α] ← PC,　M[α+1] ← PS,　PC ←α+2	Branch and Store Status
BRS α	PC ← M[α],　PS ← M[α+1]	Branch and Restore Status
HALT	停止	

意味の欄の記号の説明

M[x] :　変数名 x が指すアドレスのメモリの内容
M[α] :　ラベル α が指すアドレスのメモリの内容（α+1 等は (αのアドレス+1) のアドレス）
A :　A-レジスタ，アキュムレータ
CA :　キャリーレジスタと A-レジスタを一体として対象とする
PC :　プログラムカウンタ
PS :　プログラムステイタス
← :　内容の転送（例えば，A ← A+M[x] は，A と M[x] の内容の和を A に転送）

a=3, b=2, c = a+b

LDI	3	A←3
STO	a	M[a]←A
LDI	2	A←2
STO	b	M[b]←A
LOAD	a	A←M[a]
ADD	b	A←A+M[b]
STO	c	M[c]←A

複雑そうなプログラムの例

プログラム	意味		1度目	2度目	3度目	
LDI 0	A ← 0	A :	0			
STO a	M[a] ← A	M[a]:	0			
LOAD c	A ← M[c]	A :	3			
α: JZERO β	if(A = 0) goto β	PC :	next	next	next	β
STO c	M[c] ← A	M[c]:	3	2	1	
LOAD a	A ← M[a]	A :	0	8	16	
ADD b	A ← A+M[b]	A :	8	16	24	
STO a	M[a] ← A	M[a]:	8	16	24	
LOAD c	A ← M[c]	A :	3	2	1	
DEC	A ← A-1	A :	2	1	0	
JMP α	goto α	PC :	α	α	α	
β: HALT						停止

意味記述でのメモリの初期値：M[a] =(なんでもよい), M[b] = 8, M[c] = 3

プログラムの機能： $a = b \times c \quad (8 \times 3 = 24)$

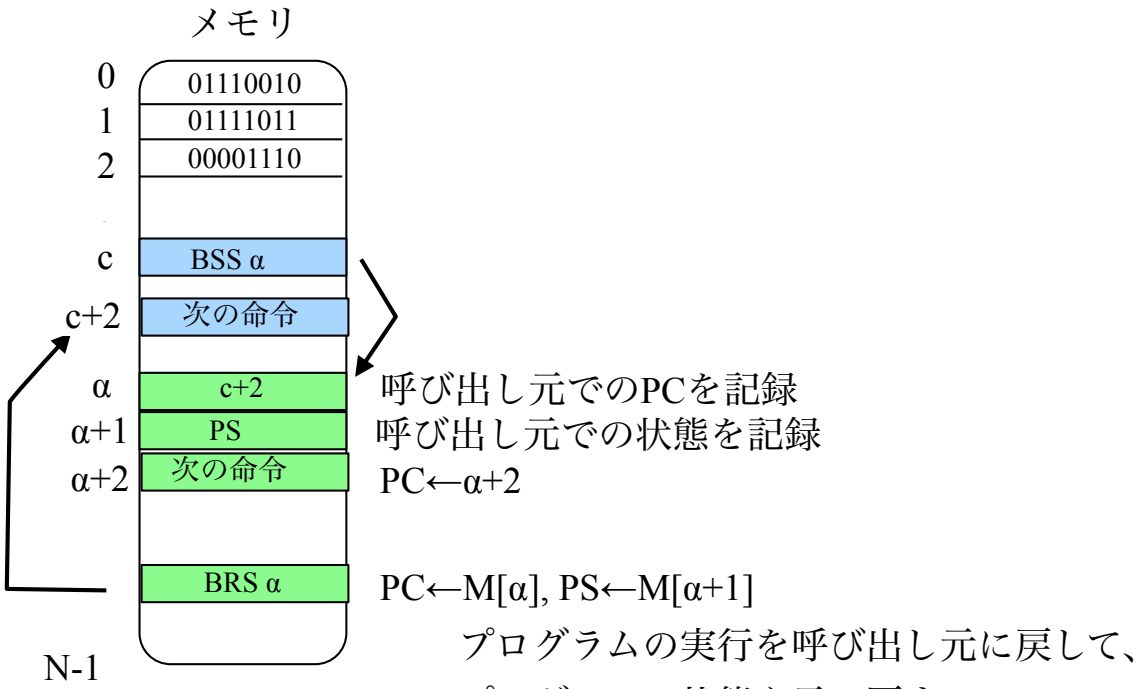
図 5.2 機械語プログラムの例

サブルーチン(関数) 呼び出し

BSS α	$M[\alpha] \leftarrow PC,\quad M[\alpha+1] \leftarrow PS,\quad PC \leftarrow \alpha+2$
BRS α	$PC \leftarrow M[\alpha],\quad PS \leftarrow M[\alpha+1]$

gosub

return



語（word）
メモリの格納場所のビット長
命令語、オペランドそれぞれ 1 語使う
(つまり命令実行によりPCは２つ増える)

サブルーチンコールを含むプログラムの例

cを1減らして0かcheck

和を取る

和を取って終了

$2x+1$

プログラム		意味		1度目	2度目	3度目	4度目
	LDI 0	A ← 0	A :	0			
	STO a	M[a] ← A	M[a]:	0			
α:	LOAD c	A ← M[c]	A :	4	3	2	1
	DEC	A ← A-1	A :	3	2	1	0
	STO c	M[c] ← A	M[c]:	3	2	1	0
	JZERO β	if(A = 0) goto β	PC :	next	next	next	β
ν:	BSS σ	gosub σ	PC :	σ+2	σ+2	σ+2	
	ADD a	A ← A+M[a]	A :	7	12	15	
	STO a	M[a] ← A	M[a]:	7	12	15	
	JMP α	goto α	PC :	α	α	α	
β:	BSS σ	gosub σ	PC :				σ+2
	ADD a	A ← A+M[a]	A :				16
	STO a	M[a] ← A	M[a]:				16
	HALT						停止
σ:	DATA		σ :	ν+2	ν+2	ν+2	β+2
	DATA		σ+1 :	some	some	some	some
	STO s	M[s] ← A	M[s]:	3	2	1	0
	ADD s	A ← A+M[s]	A :	6	4	2	0
	INC	A ← A+1	A :	7	5	3	1
	BRS σ	return	PC :	ν+2	ν+2	ν+2	β+2

注：ラベルνは意味の記述のためにつけたもので，プログラムとしては不要である．
各命令は2語とし，DATA はその場所 (1 語) にデータを入れることを意味する．

意味記述でのメモリの初期値：M[c] = 4

プログラムの機能：

図 5.3　サブルーチンコールを含む機械語プログラム