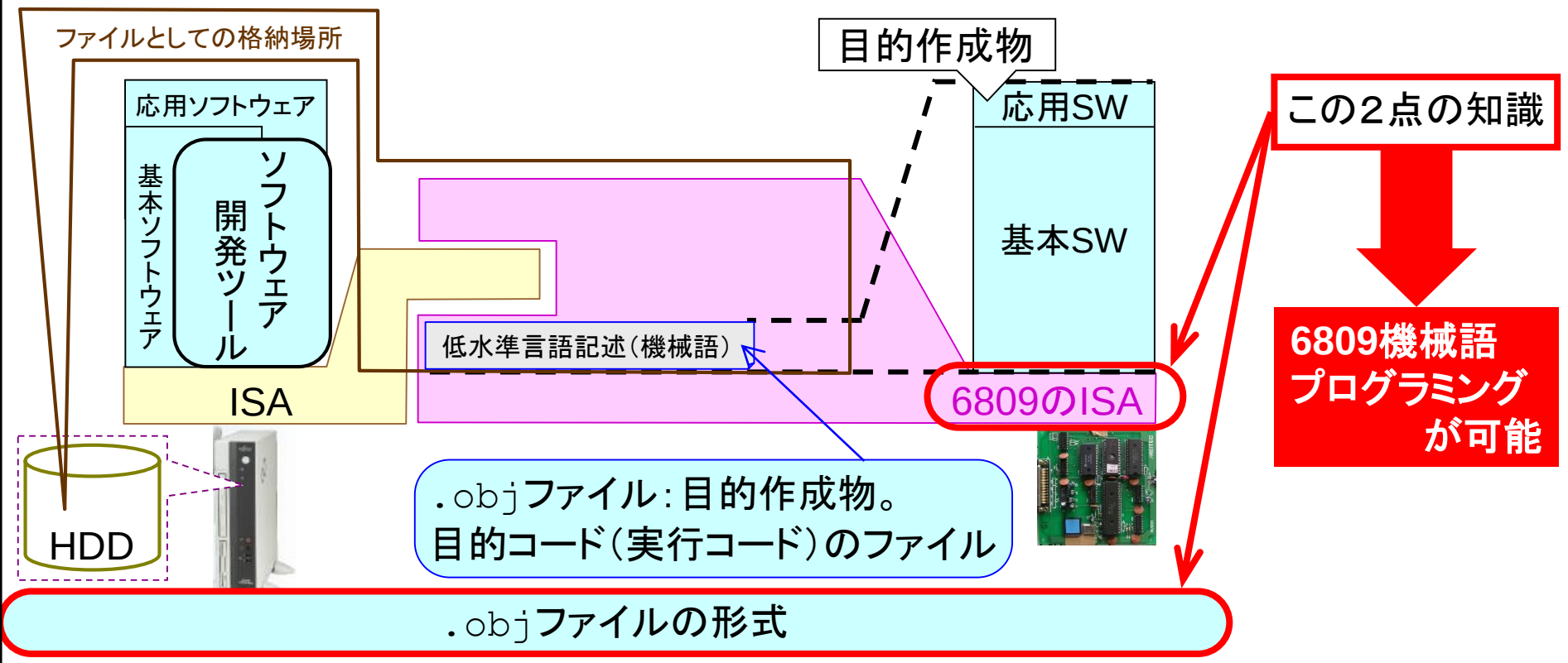
6809機械語プログラミングの特徴



有利な特徴	不利な特徴
所望の ・使用メモリサイズ ・実行サイクル数 を確実に満たすことのできる プログラミングである。	対応付けが面倒 目的の処理の各手続きに対する ・基本命令 ・使用する記憶装置 ・使用するアドレッシングモード ↔ 機械語命令 ・OPcode ・PostByte ・operand

【注】上記の特徴は、一般の機械語プログラミングにも言える

6809アセンブリ言語プログラミングのねらい

6809機械語プログラミングの
・不利な特徴の克服
・有利な特徴の保存
を可能にするプログラミングの構築

目的の処理の各手続きに対する
・基本命令
・使用する記憶装置
・使用するアドレッシングモード

対応付けの簡単化

対応付けが面倒

.asmファイル：アセンブリ言語プログラムソースファイル

アセンブラ次第
A6809K.exeの「確実」
はやや不完全

ファイルとしての格納場所

応用ソフトウェア

基本ソフトウェア

ソフトウェア開発ツール

低水準言語記述（アセンブリ言語）

アセンブラ A6809K.exe

低水準言語記述（機械語）

ISA

目的作成物

応用SW

基本SW

6809のISA

所望の
・使用メモリサイズ
・実行サイクル数
を確実に満たす
ことのできる
プログラミングである。

確実

HDD

.objファイル：目的作成物。
目的コード（実行コード）のファイル

機械語命令
OPcode／PostByte／operand

対応付けの簡単化（1/5） 面倒さ・その1

機械語operand記述時に変位計算を要する。

異なるアドレッシングモードなのに、機械語operandの記述が同じ。
⇒機械語operandを見ても、アドレッシングモードを特定できない。

Addressing Mode名			EXで使うデータの場所	（機械語）operand	備考
Immediate			operandがデータそのもの	hh / hhhh	
Inherent			参照Regにデータがある		
Direct			DP:上位8−bit＆operand:下位8−bitが示すアドレスを直接参照	hh	
Extended			operandが示すアドレスを直接参照	hhhh	A直
Indexed	Constant Offset	0-bit	Ptrが示すアドレスを直接参照		B直
		5-bit	Ptr+変位が示すアドレスを直接参照	変位はPostByteに。operand無し	C直
		8-bit		hh	D直
		16-bit		hhhh	E直
	Acc Offset		Ptr+Accが示すアドレスを直接参照		F直
	Auto Increment		Ptrが示すアドレスを直接参照⇒Ptr=Ptr＋1		G直
			Ptrが示すアドレスを直接参照⇒Ptr=Ptr＋2		H直
	Auto Decrement		Ptr=Ptr−1⇒Ptrが示すアドレスを直接参照		I直
			Ptr=Ptr−2⇒Ptrが示すアドレスを直接参照		J直
	PC Relative		PC+変位が示すアドレスを直接参照	hh / hhhh	K直
	Extended Indirection		operandが示すアドレスを間接参照	hhhh	A間
	Constant Offset	0-bit Indirection	Ptrが示すアドレスを間接参照		B間
		8-bit Indirection	Ptr+変位が示すアドレスを間接参照	hh	D間
		16-bit Indirection		hhhh	E間
	Acc Offset Indirection		Ptr+Accが示すアドレスを間接参照		F間
	Auto Increment Indirection		Ptrが示すアドレスを間接参照⇒Ptr=Ptr＋2		H間
	Auto Decrement Indirection		Ptr=Ptr−2⇒Ptrが示すアドレスを間接参照		J間
	PC Relative Indirection		PC+変位が示すアドレスを間接参照	hh / hhhh	K間
Branch Relative			PC+変位が示すアドレスを直接参照	hh / hhhh	

0-Btye

1-Byte

2-Byte

対応付けの簡単化(2/5)　面倒さ・その2

	基本命令 機能	名	OPcode
算術演算	BとXの加算	ABX	3A
	桁上げ付き加算	ADC	8種類
	加算	ADD	12種類
	0代入	CLR	5種類
	10進補正	DAA	19
	1減	DEC	5種類
	1増	INC	5種類
	乗算	MUL	3D
	符号反転	NEG	5種類
	桁上げ付き減算	SBC	8種類
	符号付ビット拡張	SEX	1D
	減算	SUB	12種類
論理演算	論理積	AND	9種類
	ビット反転	COM	5種類
	排他的論理和	EOR	8種類
	論理和	OR	9種類

	基本命令 機能	名	OPcode
データ転送	Reg間データ交換	EXG	1E
	ロード	LD	28種類
	プッシュ	PSH	34, 36
	プル(ポップ)	PUL	35, 37
	ストア	ST	21種類
	Reg間データ転送	TFR	1F
シフト	算術左シフト	ASL	5種類
	算術右シフト	ASR	5種類
	論理左シフト	LSL	5種類
	論理右シフト	LSR	5種類
	循環左シフト	ROL	5種類
	循環右シフト	ROR	5種類
比較、テスト	論理比較	BIT	8種類
	算術比較	CMP	28種類
	テスト(0減)	TST	5種類

	基本命令 機能	名	OPcode
実行順序変更	ジャンプ	JMP	0E, 7E, 6E
	サブルーチンへジャンプ	JSR	9D, BD, AD
	割込みルーチンからの復帰	RTI	3B
	サブルーチンからの復帰	RTS	39
	SW割込み	SWI	3F, 103F, 113F
その他	Regを退避し、HW割込み待ち	CWAI	3C
	実効アドレスのロード	LEA	32, 33, 30, 31
	無操作	NOP	12
	HW割込み待ち	SYNC	13

	基本命令 機能	名	OPcode
条件分岐	キャリービットがセットされていなければ分岐	BCC	24, 1024
	キャリービットがセットされていれば分岐	BCS	25, 1025
	等しいならば分岐	BEQ	27, 1027
	より以上ならば分岐(符号付きデータ比較)	BGE	2C, 102C
	より大ならば分岐　(符号付きデータ比較)	BGT	2E, 102E
	より大ならば分岐　(符号無しデータ比較)	BHI	22, 1022
	より以上ならば分岐(符号無しデータ比較)	BHS	24, 1024
	より以下ならば分岐(符号付きデータ比較)	BLE	2F, 102F
	より小ならば分岐　(符号無しデータ比較)	BLO	25, 1025
	より以下ならば分岐(符号無しデータ比較)	BLS	23, 1023
	より小ならば分岐　(符号付きデータ比較)	BLT	2D, 102D
	負数ならば分岐	BMI	2B, 102B
	等しくないならば分岐	BNE	26, 1026
	正数ならば分岐	BPL	2A, 102A
	無条件に分岐	BRA	20, 16
	分岐しない分岐	BRN	21, 1021
	サブルーチンへ分岐	BSR	8D, 17
	オーバーフロービットがセットされていなければ分岐	BVC	28, 1028
	オーバーフロービットがセットされていれば分岐	BVS	29, 1029

機械語命令は数値列なので、覚えにくい

目的の処理の各手続きに対する
- 基本命令
- 使用する記憶装置
- 使用するアドレッシングモード

- OPcode
- Post Byte
- operand

操作対象のバリエーション

機械語命令全体に操作対象のバリエーションが影響

対応付けの簡単化(3/5)　ニーモニック、アセンブリoperand

	基本命令	
	機能	名
算術演算	BとXの加算	ABX
	桁上げ付き加算	ADC
	加算	ADD
	0代入	CLR
	10進補正	DAA
	1減	DEC
	1増	INC
	乗算	MUL
	符号反転	NEG
	桁上げ付き減算	SBC
	符号付ビット拡張	SEX
	減算	SUB
論理演算	論理積	AND
	ビット反転	COM
	排他的論理和	EOR
	論理和	OR

	基本命令	
	機能	名
データ転送	Reg間データ交換	EXG
	ロード	LD
	プッシュ	PSH
	プル(ポップ)	PUL
	ストア	ST
	Reg間データ転送	TFR
シフト	算術左シフト	ASL
	算術右シフト	ASR
	論理左シフト	LSL
	論理右シフト	LSR
	左循環シフト	ROL
	右循環シフト	ROR
比較、テスト	論理比較	BIT
	算術比較	CMP
	テスト(0減)	TST

	基本命令	
	機能	名
実行順序変更	ジャンプ	JMP
	サブルーチンへジャンプ	JSR
	割込みルーチンからの復帰	RTI
	サブルーチンからの復帰	RTS
	SW割込み	SWI
その他	Regを退避し、HW割込み待ち	CWAI
	実効アドレスのロード	LEA
	無操作	NOP
	HW割込み待ち	SYNC

	基本命令	
	機能	名
条件分岐	キャリービットがセットされていなければ分岐	BCC
	キャリービットがセットされていれば分岐	BCS
	等しいならば分岐	BEQ
	より以上ならば分岐(符号付きデータ比較)	BGE
	より大ならば分岐　(符号付きデータ比較)	BGT
	より大ならば分岐　(符号無しデータ比較)	BHI
	より以上ならば分岐(符号無しデータ比較)	BHS
	より以下ならば分岐(符号付きデータ比較)	BLE
	より小ならば分岐　(符号無しデータ比較)	BLO
	より以下ならば分岐(符号無しデータ比較)	BLS
	より小ならば分岐　(符号付きデータ比較)	BLT
	負数ならば分岐	BMI
	等しくないならば分岐	BNE
	正数ならば分岐	BPL
	無条件に分岐	BRA
	分岐しない分岐	BRN
	サブルーチンへ分岐	BSR
	オーバーフロービットがセットされていなければ分岐	BVC
	オーバーフロービットがセットされていれば分岐	BVS

【注】
ニーモニック(mnemonic)
とは元来、「記憶術」の意。

ニーモニックを定義

基本命令名
に対して

目的の処理の各手続きに対する
- 基本命令
- 使用する記憶装置
- 使用するアドレッシングモード

- OPcode
- Post Byte
- operand

アセンブリoperandを定義

そのまま

Lを冠しない/冠する

必ずReg名をつける

番号をつける

場合によってReg名をつける

139のニーモニック

対応付けの簡単化（4/5） アドレッシングモードとアセンブリoperand

Addressing Mode名			EXで使うデータの場所	機械語operand	アセンブリoperand
Immediate			operandがデータそのもの	hh	#$hh
				hhhh	#$hhhh
			参照Regにデータがある		参照Reg指定
Inherent					
Direct			DP:上位8-bit＆operand:下位8-bit が示すアドレスを直接参照	hh	$hh
Extended			operandが示すアドレスを直接参照	hhhh	$hhhh
Indexed	Constant Offset	0-bit	Ptrが示すアドレスを直接参照		,Ptr / Ptr / 0,Ptr
		5-bit	Ptr+変位が示すアドレスを直接参照	変位はPostByteに。operand無し	$hh,Ptr
		8-bit		hh	$hh,Ptr
		16-bit		hhhh	$hhhh,Ptr
	Acc Offset		Ptr+Accが示すアドレスを直接参照		Acc,Ptr
	Auto Increment		Ptrが示すアドレスを直接参照⇒Ptr=Ptr＋1		,Ptr+
			Ptrが示すアドレスを直接参照⇒Ptr=Ptr＋2		,Ptr++
	Auto Decrement		Ptr=Ptr－1⇒Ptrが示すアドレスを直接参照		,-Ptr
			Ptr=Ptr－2⇒Ptrが示すアドレスを直接参照		,--Ptr
	PC Relative		PC+変位が示すアドレスを直接参照	hh / hhhh	$HHHH,PCR
	Extended Indirection		operandが示すアドレスを間接参照	hhhh	[$hhhh]
	Constant Offset	0-bit Indirection	Ptrが示すアドレスを間接参照		[,Ptr] / [Ptr] / [0,Ptr]
		8-bit Indirection	Ptr+変位が示すアドレスを間接参照	hh	[$hh,Ptr]
		16-bit Indirection		hhhh	[$hhhh,Ptr]
	Acc Offset Indirection		Ptr+Accが示すアドレスを間接参照		[Acc,Ptr]
	Auto Increment Indirection		Ptrが示すアドレスを間接参照⇒Ptr=Ptr＋2		[,Ptr++]
	Auto Decrement Indirection		Ptr=Ptr－2⇒Ptrが示すアドレスを間接参照		[,--Ptr]
	PC Relative Indirection		PC+変位が示すアドレスを間接参照	hh / hhhh	[$HHHH,PCR]
Branch Relative			PC+変位が示すアドレスを直接参照	hh / hhhh	$HHHH

区別は命令の種類で。この2つは、記述が同じなので注意。

【注】HHHH（アドレス）＝PC＋変位（hh / hhhh）　すなわち、変位（hh / hhhh）＝HHHH（アドレス）－PC　　$は基数16を明示する記号

対応付けの簡単化(5/5) アドレッシングモードとアセンブリoperand

Addressing Mode名			EXで使うデータの場所	機械語operand	アセンブリoperand
Immediate			operandがデータそのもの	hh	#$hh
				hhhh	#$hhhh
			参照Regにデータがある		参照Reg指定
Inherent					
Direct			DP:上位8-bit&operand:下位8-bitが示すアドレスを直接参照	hh	$hh
Extended			operandが示すアドレスを直接参照	hhhh	$hhhh
Indexed	Constant Offset	0-bit	Ptrが示すアドレスを直接参照		,Ptr / Ptr / 0,Ptr
		5-bit	Ptr+変位が示すアドレスを直接参照	変位はPostByteに。operand無し	$hh,Ptr
		8-bit		hh	$hh,Ptr
		16-bit		hhhh	$hhhh,Ptr
	Acc Offset		Ptr+Accが示すアドレスを直接参照		Acc,Ptr
	Auto Increment		Ptrが示すアドレスを直接参照⇒Ptr=Ptr+1		,Ptr+
			Ptrが示すアドレスを直接参照⇒Ptr=Ptr+2		,Ptr++
	Auto Decrement		Ptr=Ptr-1⇒Ptrが示すアドレスを直接参照		,-Ptr
			Ptr=Ptr-2⇒Ptrが示すアドレスを直接参照		,--Ptr
	PC Relative		PC+変位が示すアドレスを直接参照	hh / hhhh	$HHHH,PCR
	Extended Indirection		operandが示すアドレスを間接参照	hhhh	[$hhhh]
	Constant Offset	0-bit Indirection	Ptrが示すアドレスを間接参照		[,Ptr] / [Ptr] / [0,Ptr]
		8-bit Indirection			
		16-bit Indirection			
	Acc Offset Indirection		Ptr+Accが示すアドレスを間接参照		[Acc,Ptr]
	Auto Increment				
	Auto Decrement				
	PC Relative Indirection		PC+変位が示すアドレスを間接参照	hh / hhhh	[$HHHH,PCR]
Branch Relative			PC+変位が示すアドレスを直接参照	hh / hhhh	$HHHH

区別は命令の種類で。この2つは、記述が同じなので注意。

初心者のうちは、扱わなくても良いアドレッシングモード

【注】記号「$」は以降、アセンブリ言語プログラミングにて多用する。

【注】HHHH(アドレス)=PC+変位(hh / hhhh) すなわち、変位(hh / hhhh)=HHHH(アドレス)-PC $は基数16を明示する記号

6809基本アセンブリ言語文とアセンブル

目的の処理の各手続きに対する
・基本命令
・使用する記憶装置
・使用するアドレッシングモード

対応付けが面倒

対応付けの簡単化

ニーモニックを定義
　基本命令名＋（必要に応じて）Reg名など

アセンブリoperandを定義
　アドレッシングモードを記号化

・出発点は一緒
・単に対応先の表現の違い

1対1に対応

基本アセンブリ言語文

◇ ニーモニック ◇ アセンブリoperand

【注】◇は空白1つ以上。通常は1つで十分。

アセンブル

機械語命令
OPcode／PostByte／operand

F01.	OPcode				
F02.	OPcode	operand			
F03.	OPcode	operandH	operandL		
F04.	OPcode	Post Byte			
F05.	OPcode	Post Byte	operand		
F06.	OPcode	Post Byte	operandH	operandL	
F07.	OPcodeH	OPcodeL			
F08.	OPcodeH	OPcodeL	operand		
F09.	OPcodeH	OPcodeL	operandH	operandL	
F10.	OPcodeH	OPcodeL	Post Byte		
F11.	OPcodeH	OPcodeL	Post Byte	operand	
F12.	OPcodeH	OPcodeL	Post Byte	operandH	operandL

手動変換	自動変換
ハンドアセンブル	アセンブラ（A6809K.exe）
次のステップ①	次のステップ②

アセンブリ言語プログラミング

6809基本アセンブリ言語文の追加資料

基本アセンブリ言語文

◇ ニーモニック ◇ アセンブリoperand 【注】◇は空白1つ以上。通常は1つで十分。

- man6809.pdf
  （p.2-1～p.2-6、p.D-1～p.D-4、p.F-3/F-4）
- CAE_AddressingModes_V2.pdf

- man6809.pdf
  （p.A-1～p.A-72、
  p.C-2～p.C-4、
  p.D-1～p.D-4）

- 6809_Insn_MainList-AL.pdf 6809_Insn_MainList-FULL.pdf
- 6809_Insn_Branch-AL.pdf 6809_Insn_Branch-FULL.pdf
- 6809_Insn_PshPul-AL.pdf 6809_Insn_PshPul-FULL.pdf

資料は全て、講義用Webページからも開ける。

練習

(1)

```
LDA #$03
ADDA #$08
STA $1008
RTS
```

(2)

```
LDA #$68
STA $1100
INCA
STA $1101
LDX $1100
STX $1102
RTS
```

上の2つの基本アセンブリ言語文の列、それぞれについて、
（a）ハンドアセンブルせよ。（RTSも含む）
（b）プログラムの内容を説明せよ。（RTSを除く）

6809アセンブリ言語プログラミング

6809アセンブリ言語プログラミング

6809機械語プログラミングの
・不利な特徴の克服
・有利な特徴の保存
を可能にするプログラミングの構築

アルファベットの
大文字、小文字の
区別をしない。

.asmファイル:
アセンブリ言語プログラムソースファイル

対応付けの簡単化

- 基本アセンブリ言語文
  - ◇ ニーモニック ◇ アセンブリoperand
- アセンブラの用意『A6809K.exe』

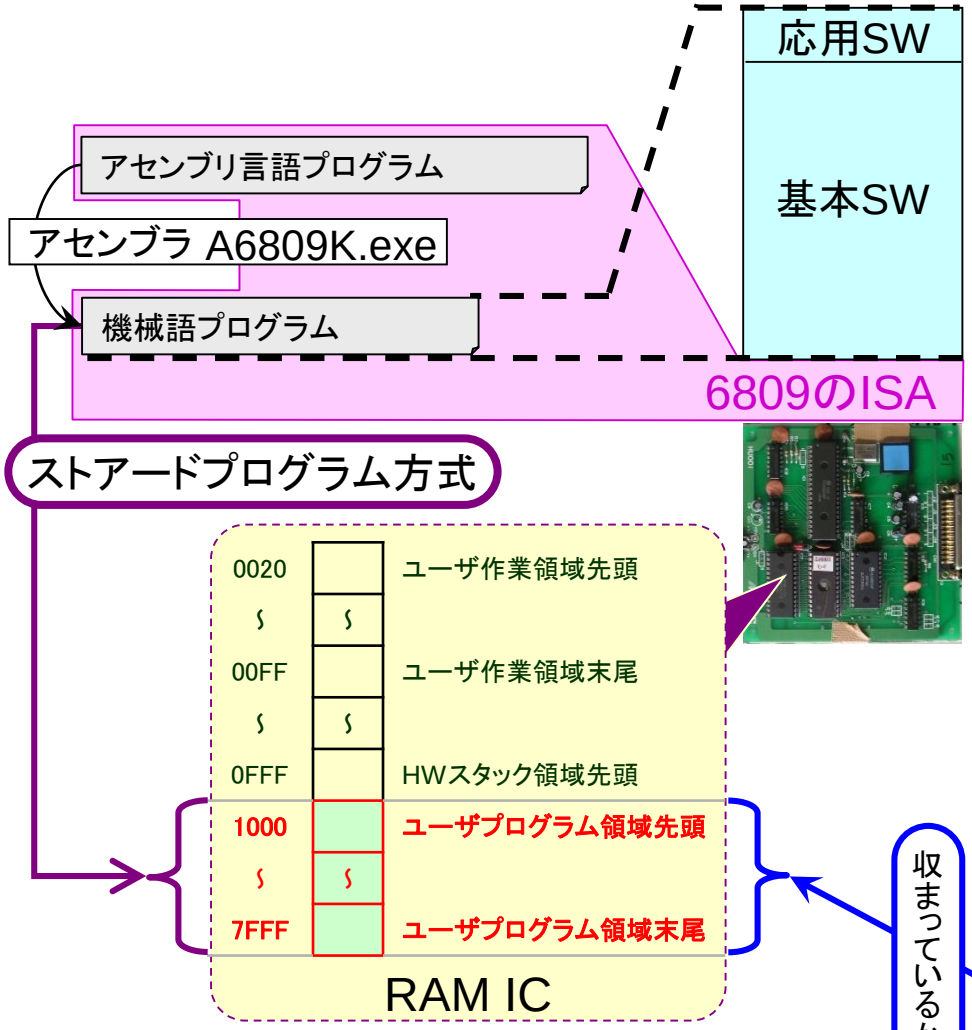
- 擬似命令（Pseudo-instruction）
  アセンブラに対する命令（指示）。
  アセンブラディレクティブ
  （Assembler Directive）とも呼ばれる。

- プログラミングの利便性の向上
  （アセンブラが対応している必要がある）
  - ラベル
  - コメント
  - マルチステートメント
  - ~~マクロ~~ 本科目のアセンブラは非対応

  etc…

【注】アセンブリ言語プログラムの、最後に記述するニーモニックは必ず「RTS」とする。
・・・機械語命令「39」に対応するニーモニック。モニタプログラムとの関係による。

6809アセンブリ言語プログラミング　擬似命令　その1



●プログラムの転送先アドレスの先頭addrB、末尾addrEの指定方法

■機械語プログラム(.obj)

.obj形式にて自然に指定される。

addrB:1000　　addrE:1016

```
@1000[8603121212121212121212121212121212]
@1010[121212B7101739]
E
```

■アセンブリ言語プログラム(.asm)

addrB:擬似命令「ORG」で明示する。

addrE:擬似命令「END」をアセンブラが解釈し自動的に定まる。

addrB:1000

addrE:自動的に定まる

(人には分からないので要確認)

```
ORG $1000
LDA #$03
NOP
 ⋮   } 17個
NOP
STA $1017
RTS
END
```

擬似命令	文の書式	役割
ORG	◇ORG◇$hhhh	プログラムの転送先の先頭アドレス(hhhh)を指定
END	◇END	プログラムの末尾を指定

6809アセンブリ言語プログラミング　擬似命令　その2

●プログラム中で扱うデータの記憶領域を確保する方法

■機械語プログラム(.obj)
.obj形式にて確保する。

```
@1000[F6100AFB100BB7100C390308]
E
```

■アセンブリ言語プログラム(.asm)　以下の擬似命令を用いて確保する。

擬似命令	文の書式	役割
FCC	◇FCC◇/strings/	文字列“strings”のASCIIコード列の記憶領域確保
FCB	◇FCB◇$hh	1バイト定数$hhの記憶領域確保
FDB	◇FDB◇$hhhh	2バイト定数$hhhhの記憶領域確保

《例》

```
ORG $1000
LDB $100A
ADDB $100B
STB $100C
RTS
FCB $03
FCB $08
END
```

《例》

```
ORG $1000
LDB $100A
ADDB $100B
STB $100C
RTS
FDB $0308
END
```

【注】 上の2つのアセンブリ言語プログラム例は、同じ振る舞いをする

6809アセンブリ言語プログラミング　利便性向上　ラベル

◆ラベル付きアセンブリ言語文

ラベル 基本アセンブリ言語文

◆ラベル付き擬似命令その2文

ラベル 擬似命令その2文

◆ラベルのみ

ラベル

ラベル：
アドレスを記号化記述したもの。
アセンブラは、
ラベルを実アドレスとみなし、
機械語プログラムを生成する。

【注】左の書式のそれぞれの先頭に空白はない。

《ラベルの利用例》

```
 ORG $1000
 LDB $100A
 ADDB $100B
 STB $100C
 RTS
 FCB $03
 FCB $08
 END
```

```
 ORG $1000
START LDB LIN1
 ADDB LIN2
 STB LOUT
 RTS
LIN1 FCB $03
LIN2 FCB $08
LOUT
 END
```

練習

下記のアセンブリ言語プログラムにおいて、
格納データ$03を取り出し、それを格納データ$04に上書きしたい。
このとき、
（1）hhhh，HHHHの値をいくつにすべきか求めよ。
（2）hhhh，HHHHの値を求めなくて済むように、
　　ラベルを利用してプログラムを書き直せ。

```
ORG $1000
LDA $hhhh
STA $HHHH
RTS
FCB $03
FCB $04
END
```

6809アセンブリ言語プログラミング　利便性向上　コメント

◆コメント文（一行）

＊ 任意文字列 【注】先頭に空白はない。

アセンブラは、
行末までをコメントとみなし、
読み飛ばす。

6809アセンブリ言語プログラミング　擬似命令　その3

●プログラムの記述に便利な記号の定義をする方法

■機械語プログラム(.obj)　特別な記号は扱えない。

【例】x=03, y=08 に対する2x＋3yをM[10A0]:M[10A1]に格納

```
@1000[4FC603CB03F710B08900C608CB088900]
@1010[CB08F710A18900F610B0FB10A1F710A1]
@1020[8900B710A039]
E
```

数値	03	08	10A0	10A1	10B0
登場個数	2	3	1	3	2

x、y、格納場所等の
変更は簡単？

■アセンブリ言語プログラム(.asm)　以下の擬似命令を用いて定義する。

擬似命令	文の書式	役割
EQU	MYSYM□EQU□$hhhh	記号“MYSYM”を、値$hhhhとして、定義する

```
 org $1000
NUM1 equ $03
NUM2 equ $08
LCAR equ $10A0
LANS equ $10A1
LTMP equ $10B0
 CLRA
```

```
 LDB #NUM1
 ADDB #NUM1
 STB LTMP
 ADCA #$00
 LDB #NUM2
 ADDB #NUM2
 ADCA #$00
```

```
 ADDB #NUM2
 STB LANS
 ADCA #$00
 LDB LTMP
 ADDB LANS
 STB LANS
 ADCA #$00
```

```
 STA LCAR
 RTS
 END
```

6809アセンブリ言語プログラミング　利便性向上　マルチステートメント

◆マルチステートメント

文 : 文 : ・・・・・ : 文 : 文

1行に複数の文を記述する方法。
（通常は、1行に1つの文を記述する。）
文と文の間をコロン「:」で区切る。

上の各「文」は、下の種類のいずれか。特に、文頭の空白に注意

文の種類	文頭の空白
基本アセンブリ言語文	あり
擬似命令その1文	あり
擬似命令その2文	あり
ラベル付きアセンブリ言語文	なし
ラベル付き擬似命令その2文	なし
ラベルのみ	なし
コメント文	なし
擬似命令その3文	なし

【注】 配布ファイル中の3つの.asmは、この記述方法を用いていることに留意。

6809アセンブリ言語プログラミング　まとめ(1/2)

6809アセンブリ言語プログラミング

6809機械語プログラミングの
・不利な特徴の克服
・有利な特徴の保存
を可能にするプログラミングの構築

アルファベットの
大文字、小文字の
区別をしない。

.asmファイル:
アセンブリ言語プログラムソースファイル

- 基本アセンブリ言語文
  - ◇ ニーモニック ◇ アセンブリoperand
- アセンブラの用意『A6809K.exe』
- 擬似命令(Pseudo-instruction)
  アセンブラに対する命令(指示)。
  アセンブラディレクティブ
  (Assembler Directive)とも呼ばれる。
- ◆プログラミングの利便性の向上
  (アセンブラが対応している必要がある)
  - ・ラベル
  - ・コメント
  - ・マルチステートメント

【注】アセンブリ言語プログラムの、最後に記述するニーモニックは必ず「RTS」とする。
・・・機械語命令「39」に対応するニーモニック。モニタプログラムとの関係による。

6809アセンブリ言語プログラミング　まとめ(2/2)

6809アセンブリ言語プログラミング

6809機械語プログラミングの
・不利な特徴の克服
・有利な特徴の保存
を可能にするプログラミングの構築

アルファベットの
大文字、小文字の
区別をしない。

.asmファイル:
アセンブリ言語プログラムソースファイル

文の種類	文頭の空白
基本アセンブリ言語文	あり
擬似命令その1文	あり
擬似命令その2文	あり
ラベル付きアセンブリ言語文	なし
ラベル付き擬似命令その2文	なし
ラベルのみ	なし
コメント文	なし
擬似命令その3文	なし

【注】アセンブリ言語プログラムの、最後に記述するニーモニックは必ず「RTS」とする。
・・・機械語命令「39」に対応するニーモニック。モニタプログラムとの関係による。

練習

(1)

```
LDA #$03
ADDA #$08
STA $1008
RTS
```

(2)

```
LDA #$68
STA $1100
INCA
STA $1101
LDX $1100
STX $1102
RTS
```

前にハンドアセンブルを行った、
上の2つの基本アセンブリ言語文の列、それぞれについて、
以下のステップを進めよ。

（a） 擬似命令やラベルなどを適宜用いて、アセンブリ言語プログラムを作成
（b） A6809K.exeでアセンブル
（c） 前にハンドアセンブルした結果と比較
（d） 09brdまたはsim09brdで実行
（e） **09brdでの実行結果**をコピーし、
ワープロソフトに『**MSゴシック**』フォントで貼り付け
（f） 実行結果の確認と考察
（g） プログラムの使用メモリサイズと実行サイクルの算出