情報理工学実験 A・情報通信実験 A 報告書

項目名	マイクロコンピュータ

実験実施日		Ⅰ	2025 年　10 月　7 日		
		Ⅱ	2025 年　10 月　14 日		
場所	63-B1-03,26 室		時間	3－4 時限	
天候	曇り 曇り	室温	24℃ 24℃	湿度	49% 49%

早稲田大学基幹理工学部		学科名	情報通信学科
学年	2	班	1
学籍番号	1W242038-6		
氏名	植木敬太郎		
共同実験者	大里太陽		

1 章：実験報告内容の概要

本実験は，教育用 8 ビットマイクロプロセッサ(Kue-chip2)を対象にアセンブリ言語でのプログラミングを行い，命令が読みだされ，解読され，実行されるという基本的なサイクルを確認した。これにより，計算機の仕組みを単に理論として理解するだけでなく，実際の操作を通してソフトウェア的な観点とハードウェア的な観点の双方から学ぶことを目的とした。

2 章：結果

2.1 加算プログラムのトレース結果（実験 3.2）

ADD 命令実行中の各種レジスタのトレースを表 2.1 に，ADC 命令実行中のフラグレジスタのトレースを表 2.2 に示す。

表 2.1 ADD 命令実行中の各種レジスタのトレース

演算	命令	機械語	各種レジスタ		P0	P1	P2	P3	P4
126+ 2	LD ACC, (100h)	65 00	プログラムカウンタ	PC	01	01	02	02	02
			フラグレジスタ	FLAG	00	00	00	00	00
			アキュムレータ	ACC	00	00	00	00	7E
			メモリアドレスレジスタ	MAR	00	00	01	00	00
			命令レジスタ	IR	00	65	65	65	65
	ADD ACC, (101h)	B5 01	プログラムカウンタ	PC	03	03	04	04	04
			フラグレジスタ	FLAG	00	00	00	00	06
			アキュムレータ	ACC	7E	7E	7E	7E	80
			メモリアドレスレジスタ	MAR	02	02	03	01	01
			命令レジスタ	IR	65	B5	B5	B5	B5
	ST ACC, (102h)	75 02	プログラムカウンタ	PC	05	05	06	06	06
			フラグレジスタ	FLAG	06	06	06	06	06
			アキュムレータ	ACC	80	80	80	80	80
			メモリアドレスレジスタ	MAR	04	04	05	02	02
			命令レジスタ	IR	B5	75	75	75	75

表 2.2 ADC 命令実行中のフラグレジスタのトレース

演算	命令	機械語	各種レジスタ		P0	P1	P2	P3	P4
2+(-3)	ADC ACC, (101h)	95 01	フラグレジスタ	FLAG	00	00	00	00	02
126+(-126)					00	00	00	00	09
-127+(-2)					00	00	00	00	0C
2+ 3					00	00	00	00	00
126+ 1					00	00	00	00	00

2.2 与えられた課題の内容（グループ別課題）

2 台の Kue-chip を便宜上 Kue-chip A, Kue-chip B とする。Kue-chip A, Kue-chip B にそれぞれ任意の 5 つのデータを格納する。
どちらか一方の 5 つのデータをもう片方に送り，バブルソートによりデータを昇順(または降順)に整列する。大きい方から 5 つ目までのデータを Kue-chip A の 100h～104h 番地に，小さい方の 5 つのデータを Kue-chip B の100h～104h番地にそれぞれ格納せよ。

2.3 グループ別課題の解法、フローチャート、プログラム

まず，Kue-chip A, Kue-chip B(以下 A, B とする)それぞれに 5 つのデータを格納し，B が自らのデータを A に転送した。これにより，A の側には A と B あわせた合計 10 個のデータが揃うことになる。A に格納された 10 個のデータを対象にバブルソートを用いた整列処理を実行した。比較と交換は隣接要素間で行い，右側の値が左側より大きい場合に入れ替えることにより，大きな値が順次左側に移動する降順ソートを実現した。バブルソート後，A は上位 5 個のデータを自身のメモリ領域100h～104hに保存した。残りの下位 5 個のデータは B に転送され，自身のメモリ領域100h～104hに保存した。以上より，A には大きい方から 5 個のデータが，B には小さい方から 5 個のデータがそれぞれ格納される結果が得られた。

図 2.1, 2.2 に A, B それぞれのフローチャート，表 2.3, 2.4 にプログラムを示す。また，表 2.5 は，入力と出力の実行例である。

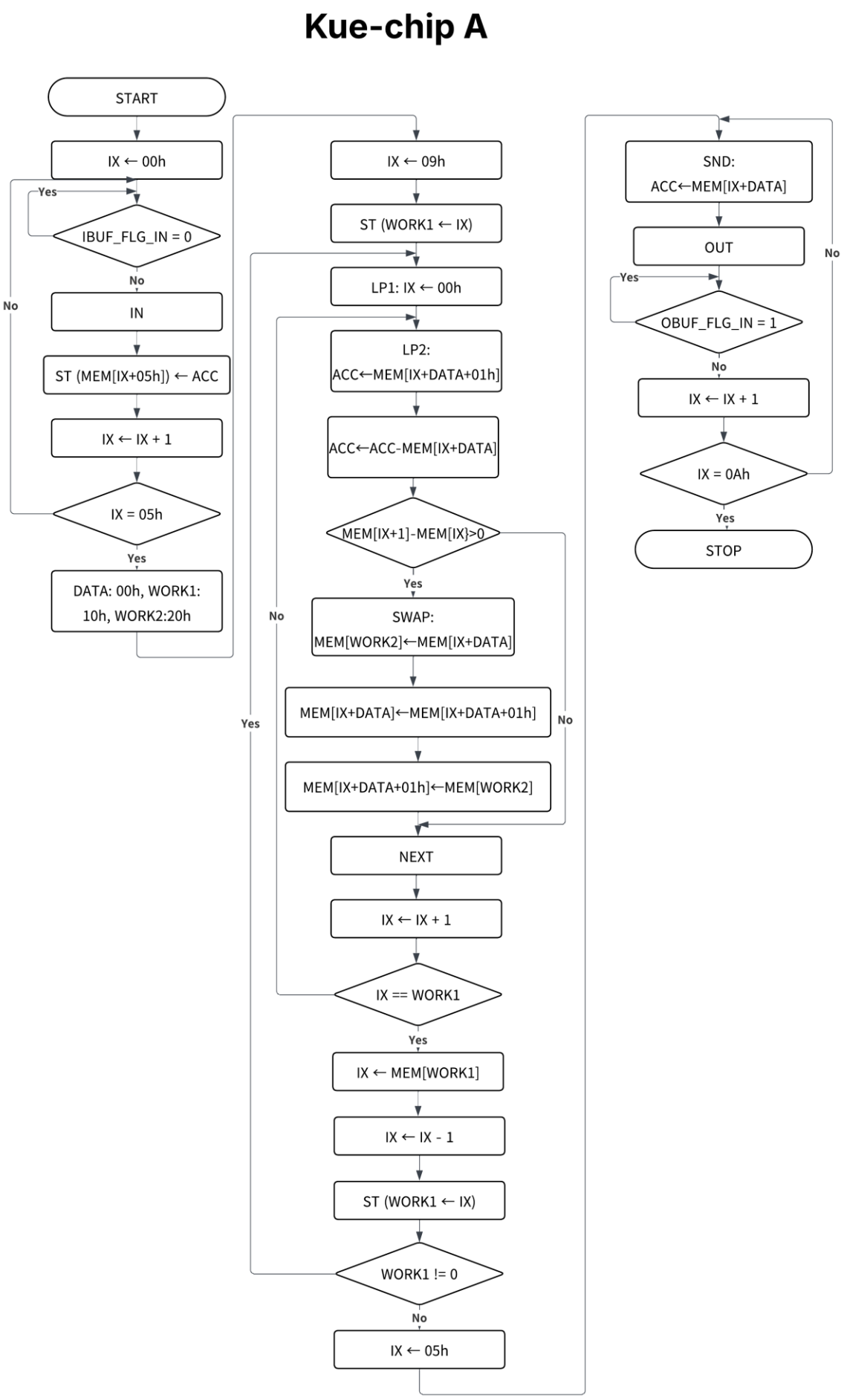

図 2.1 Kue-chip A のフローチャート

Kue-chip B

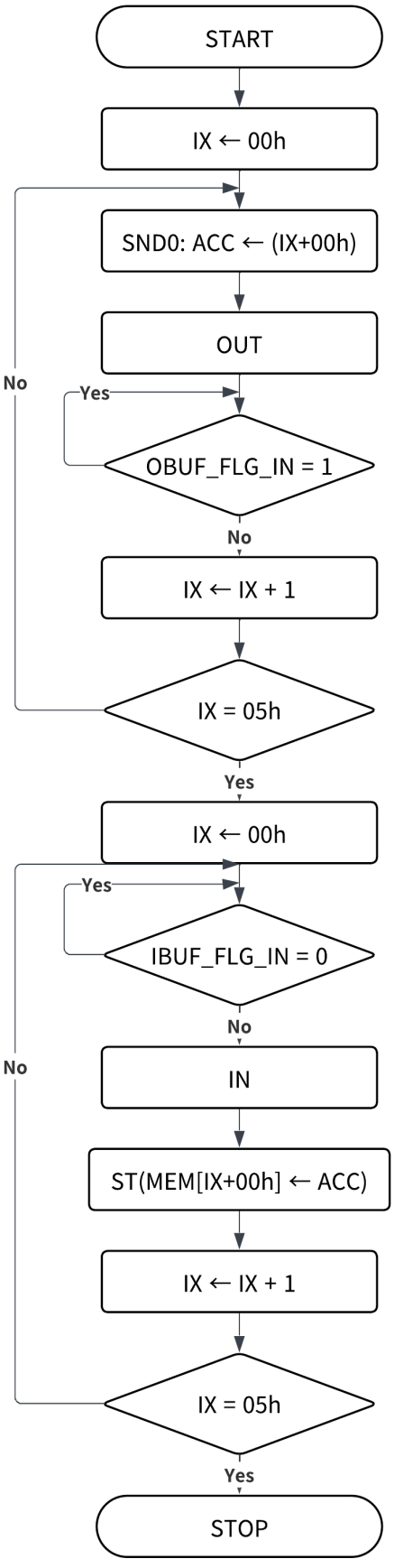

図 2.2 Kue-chip B のフローチャート

表 2.3 グループ別課題のプログラム(Kue-chipA)

アドレス	データ(機械語)	アセンブリ言語		
		ラベル	命令	オペランド
00	6A 00		LD	IX, 00H
02	34 02	NI1	BNI	NI1
04	1F		IN	
05	77 05		ST	ACC, (IX+05H)
07	BA 01		ADD	IX, 01H
09	FA 05		CMP	IX, 05H
0B	31 02		BNZ	NI1
0D	6A 09		LD	IX, 09H
0F	7D 10		ST	IX, (WORK1)
11	6A 00	LP1	LD	IX, 00H
13	67 01	LP2	LD	ACC, (IX+DATA+01H)
15	A7 00		SUB	ACC, (IX+DATA)
17	33 1B		BP	SWAP
19	30 27		BA	NEXT
1B	67 00	SWAP	LD	ACC, (IX+DATA)
1D	75 20		ST	ACC, (WORK2)
1F	67 01		LD	ACC, (IX+DATA+01H)
21	77 00		ST	ACC, (IX+DATA)
23	65 20		LD	ACC, (WORK2)
25	77 01		ST	ACC, (IX+DATA+01H)
27	BA 01	NEXT	ADD	IX, 01H
29	FD 10		CMP	IX, (WORK1)
2B	31 13		BNZ	LP2
2D	6D 10		LD	IX, (WORK1)
2F	AA 01		SUB	IX, 01H
31	7D 10		ST	IX, (WORK1)
33	31 11		BNZ	LP1
35	6A 05		LD	IX, 05H
37	67 00	SND	LD	ACC, (IX+DATA)
39	10		OUT	
3A	3C 3A	NO1	BNO	NO1
3C	BA 01		ADD	IX, 01H
3E	FA 0A		CMP	IX, 0AH
40	31 37		BNZ	SND
42	0F		HLT	

表 2.4 グループ別課題のプログラム(Kue-chipB)

アドレス	データ(機械語)	アセンブリ言語		
		ラベル	命令	オペランド
00	6A 00		LD	IX, 00H
02	67 00	SND0	LD	ACC, (IX+00H)
04	10		OUT	
05	3C 05	W0	BNO	W0
07	BA 01		ADD	IX, 01H
09	FA 05		CMP	IX, 05H
0B	31 02		BNZ	SND0
0D	6A 00		LD	IX, 00H
0F	34 0F	RB	BNI	RB
11	1F		IN	
12	77 00		ST	ACC, (IX+00H)
14	BA 01		ADD	IX, 01H
16	FA 05		CMP	IX, 05H
18	31 0F		BNZ	RB
1A	0F		HLT	

表 2.5 グループ別課題の実行例

アドレス	データ(機械語)			
	入力		出力	
	Kue-chip A	Kue-chip B	Kue-chip A	Kue-chip B
100	35	44	91	24
101	91	56	56	21
102	13	32	44	13
103	11	21	35	11
104	24	10	32	10

3 章：考察

本章では，ADD 命令実行中の各種レジスタのトレース，ADD 命令と ADC 命令の相違，グループ別課題に関する評価と改善点に分けて考察を行う。

3.1 ADD 命令実行中の各種レジスタのトレースについて

本節では，表 2.1 に示したトレース結果を参考に，ADD 命令の実行過程における各レジスタの挙動をフェーズごとに確認する。まず，各フェーズで生じる処理の概要を以下に示す。

(1) P0：すべての命令に共通し，「PC の内容が MAR に転送された後，PC がインクリメントされる。」[1)]

(2) P1：「MAR で指定されたアドレスの内容が命令レジスタ(IR)に読み込まれる。」[1)]

(3) P2：LD 命令・ST 命令，または第 2 オペランドがレジスタ以外である ADD 命令の場合，「PC の値が再び MAR に転送され，PC がさらに 1 加算される。」[1)]

(4) P3：本実験では，LD 命令，ADD 命令，および ST 命令においてオペランドに絶対アドレス（100h，101h，102h）を指定した。そのため，「MAR はまず命令語に含まれるアドレス値を保持し，次にそのアドレスが示すメモリ内容を再度 MAR に格納する動作を行った。」[1)]

(5) P4：命令ごとに処理が分岐する。

- LD 命令：第 1 オペランドで指定されたレジスタに，MAR の指すメモリ値が書き込まれる。
- ST 命令：第 1 オペランドのレジスタ値が，MAR で指定されたメモリに保存される。
- ADD 命令：ALU で加算が実行され，その結果が第 1 オペランドに反映される。同時に演算で発生した各種フラグがフラグレジスタに更新される。

続いて，表 2.1 に基づき，各レジスタの具体的な挙動を確認する。

3.1.1 プログラムカウンタ (PC)

表 2.1 のトレース結果から，PC は P0 と P2 で MAR へ転送され，その直後にインクリメントされていることが確認された。これは 2 バイト命令である ADD の処理が途切れることなく行われていることを示す。

3.1.2 アキュムレータ (ACC) とフラグレジスタ (FLAG)

命令実行中，演算の結果を保持する ACC は00h → 7Eh → 80hと推移した。初期状態では00hにクリアされており，ロード命令でメモリ100hに格納されていた7Ehが代入された。続いて ADD 命令によってメモリ101hに格納されていた02hが加算され，結果として80hとなった。この結果は演算ユニット ALU が仕様どおりに動作していることを示している。また計算結果に応じて更新される FLAG の値は，ADD 命令実行後に06hに更新された。ビットごとの解析では NF=1, ZF=0, CF=0, VF=0 である。「ここで，CF は桁上げの有無を示し，VF は符号付き演算における桁あふれの有無を示す。また，NF は結果が負数であるかどうかを示し，ZF は結果が 0 であるかどうかを示す。」[1)]

■ NF = 1

80h = $1000\ 0000_{(2)}$であり，最上位ビットに 1 が立っている。2 の補数表現では，最上位ビットが 1 である場合に負数を意味するため，NF が 1 にセットされている。

■ ZF = 0

演算結果は 80h であり，0 とは異なるためゼロフラグは 0 である。

■ $CF = 0$

CF は無符号加算において，8 ビットを超えて 9 ビット目への桁上がりが発生した場合に 1 となる。今回の演算は$7Eh\ (126) + 02h\ (2) = 80h\ (128)$ であり，結果は0〜255の範囲内に収まっているため，桁上がりは生じていない。したがって$CF = 0$となる。

■ $VF = 0$

桁あふれフラグは符号付き演算において，同符号同士の加算の結果，逆符号の値が得られた場合に 1 となる。今回の計算では$126(0111\ 1110_{(2)}) + 2(0000\ 0010_{(2)}) = 128(1000\ 0000_{(2)})$となり，符号ビットは 1 に変化して負数と解釈される。しかし，Kue-chip2 の仕様上この結果は桁あふれとは判定されず，VF は 0 のままである。

3.1.3 メモリアドレスレジスタ(MAR)

MAR は ADD 命令が 2 バイト形式であることにより，P0 で命令語のアドレスを，P2 でオペランドのアドレスを保持していた。観測結果からは，MAR が逐次的に更新され，対応するメモリ位置からオペコードやオペランドを適切に読み出していたことが確認された。

3.1.4 命令レジスタ(IR)

現在実行中の命令を保持する IR は命令が読みだされるごとに更新され，トレースでは$65h \rightarrow B5h \rightarrow 75h$と推移した。これは P1 において，MAR が示すメモリの位置を正しく取り込んでいることを示す。したがって，IR が正常に機能していることが確認された。

3.2 ADD 命令と ADC 命令の違い

続いて，表 2.2 のトレース結果に基づき，ADC 命令と ADD 命令の相違点について考察する。この 2 つの命令は，オペランド間の加算処理を行うという点で共通する。しかし，ADC 命令は ADD 命令とは異なり，桁上げフラグ(CF)を演算に入力として取り込む。また，ADC 命令は，演算結果に応じて CF，VF，NF，ZF を更新する。

■ ZF と CF のセット$126 + (-126)$の結果

演算結果が0となり，ZF が 1 にセットされたことが$09h = 1001_{(2)}$からわかる。同時に CF も 1 にセットされている。これは，8 ビット 2 の補数に基づく演算において，最高位からの桁上げが発生する(式 3.1)という ADC 命令の特性を示すものである。

$$11111110_{(2)} + 10000010_{(2)} = 1\ 00000000_{(2)} \tag{3.1}$$

■ VF と CF のセット$-127 + (-2)$の結果

$0Ch = 1100_{(2)}$は，$CF = 1$と$VF = 1$を示している。VF は，負数同士の加算を行った際，結果が 8 ビット符号付き整数で表現できる範囲$[-128(10000000_{(2)}), +127(01111111_{(2)})]$を超えたために 1 にセットされている。また，CF については

$$10000001_{(2)} + 11111110_{(2)} = 1\ 01111111_{(2)} \tag{3.2}$$

という演算により 9 ビット目に桁上がりを生じたため 1 にセットされた。

ここで重要なのは，符号付き演算と無符号演算とで参照すべきフラグが異なるという点である。符号付き演算（2 の補数表現）においては，桁あふれの検出は VF によって行うことができる。VF は，正の数同士の加算で負の値が得られる，あるいは負の数同士の加算で正の値が得られるといった符号の不整合を適切に反映するため，ADD と ADC のいずれを用いた場合でも結果に大きな差は生じない。これに対し，無符号演算では VF は意味を持たず，範囲外（256 以上）の発生は CF のみによって検出される。実験結果においても，ADD 命令では桁上がりが CF に反映されない場合が確認され

た一方，ADC 命令では CF が正しく更新されることが示された。このことから，無符号演算においては ADC を用いることで桁あふれを確実に検出できる点に意義があると結論づけられる。

■ NF のセット$2 + (-3)$の結果

$02\mathrm{h} = 0010_{(2)}$は，NF が 1 にセットされたことを示しており，結果が負数であったことを裏付けている。

3.2.3 多倍長演算

ADC 命令の CF を演算の入力として利用する処理は，16 ビット以上の多倍長演算を可能にする仕組みであることがわかった。「Kue-chip2 は 8 ビットマイクロプロセッサ」[1]であるため，16 ビット以上の数値を扱う際は，下位バイトの演算で発生した桁上がりを，上位バイトの演算に持ち越さなければならない。したがって，ADC 命令は，ADD 命令が提供する基本的な加算機能に加え，フラグレジスタの状態を参照し，計算機が処理できるデータ幅を超える算術演算を可能にする命令であると結論づけられる。

3.3 グループ別課題について

3.3.1 バブルソートの実装

表 2.3 に示した Kue-chip A のプログラムでは，受信データを自身の配列と連結した後，バブルソートを適用した。比較は SUB によって隣接要素の差を算出し，結果が正のときに BP 命令で分岐して交換処理を行うことで，大きい値を先頭側へ移動させる方式となっている。すなわち本実装は降順ソートに相当し，10 要素全体の完全な整列が実現されている。

3.3.2 本実験で実装したバブルソートの改善点

一般にバブルソートは最悪計算量が$O(n^2)$である。したがって，配列の要素数が増えると処理効率が著しく低下する。しかし，各パスで「交換が一度でも発生したか」を記録する早期終了判定フラグ(SWAPFLAG)を導入することでこの問題を改善できる。具体的な導入方法は以下の通りである。

・各外側ループ開始時に SWAPFLAG を 0 にし，整列対象の配列において交換が行われた場合に SWAPFLAG を 1 に設定する。

・内側ループ終了後，SWAPFLAG の値が 0 であれば，すべての要素がソート済みになるため，処理を終了する。

この実装により，完全に整列済みの場合計算量は$O(n)$である。また，部分的に整列している場合でも余分なループを省略することが可能である。

3.3.3 本実装の評価

本実験のグループ別課題は配列の規模が 10 個であったため十分現実的であったが，より大規模なデータを扱う際には 3.3.2 で述べた早期終了判定フラグを導入することが効果的であると考察できる。これにより，計算量の削減と実行効率の向上が期待できる。

4 章：まとめ

本実験で扱った Kue-chip2 は，教育用に設計された簡易的なマイクルプロセッサであるため，内部動作を追いやすく，各命令の処理やレジスタの挙動を観察しながら学習できる点が大きな利点であった。一方で，レジスタの数が限られていることから，複雑なアルゴリズムや大規模処理を実装することは難しい。

参考文献

1) 早稲田大学基幹理工学部 情報通信学科, 情報通信実験 A テキスト, 電気工学実験室編, 19-49, 2025