I　サンプルプログラム 7.1

i　SSC で計算可能な加数、被加数の範囲はどこまでか。

8bit の 2 の補数表現で表現可能な-2^7から2^7-1まで、-128 から 127 に収まるような数字。

同符号の足し算の場合はオーバーフローの可能性があるが、異符号の 2 数字の加算の場合と同符号の減算の場合はオーバーフローの可能性はない。

ii　オーバーフローレジスタが点灯するのは、どんな場合でしたか。

正の値同士の足し算で、結果が 128 以上の場合。

負の値同士の足し算で、結果が-128 未満の場合。

負の値を入力した場合。（メモによる。とはいえ、本当にそうであったのかは若干の疑問あり。）

iii　期待通りの動作をさせるための修正。

負の値を入力すると繰り返しての計算が行えないため次のとおりに修正を行った。

番地	機械語	ニモニック表記	コメント
0	000　00001	Jump　1	;　1 番地にジャンプする（無意味な命令）
1	101　01001	Read　9	;　トグルスイッチのデータを読んで 9 番地に格納
2	101　01010	Read　10	;　トグルスイッチのデータを読んで 10 番地に格納
3	011　01010	Load　10	;　10 番地の内容を Accumulator へ
4	001　01001	Add　9	;　9 番地の内容を Accumulator の内容へ加算する
5	100　01010	Store　10	;　和（Accumulator の内容）を 10 番地に格納
6	110　01010	Write　10	;　10 番地の内容を OUTPUT REGISTER へ出力・表示
7	011　00000	Load　0	;　0 番地のデータを読んで Accumulator へ格納（常に Jump を成立させるため）
8	000　00001	Jump　1	;　1 番地にジャンプする
9	000　00000	0	;　読み込んだデータの保存場所
10	000　00000	0	;　読み込んだデータと計算結果の保存場所

無駄な気もするが、Jump 命令の直前に 0 番地の値を読み込んで Accumulator が必ず正の値を持つようにした。それに従って Read 命令でデータを保存する場所が変化している。

II　サンプルプログラム 7.2

i　じっこうしてみると'10000000'まで表示したところで停止します。その理由はなんでしょうか。

5 番地にある Jump 命令が、'10000000'が Accumulator に入ることで動作せず、ループから抜けだしてしまうから。

ii　'11111111'まで表示を繰り返すように修正してください。

サンプルプログラムの 8 番地までのループで2^7回の表示ができている。もう2^7回ループを回すことで表示が終了するのでその分のカウンタを別途用意して回すことで'11111111'まで表示を行うようなプログラムを組んだ。ただ、このプログラムではすべて表示し終わるまでに 30 分以上かかるのが問題。軽い方法はないものかと思ったが思いつかなかった。

番地	機械語	ニモニック表記		コメント
0	000　00001	Jump	1	;　1 番地にジャンプする（無意味な命令）
1	011　01011	Load	11	;　11 番地のデータを読み込んで Accumulator に格納
2	001　00000	Add	0	;　Accumulator に 0 番地(1)を加える。
3	100　01011	Store	11	;　Accumulator の内容を 11 番地に保存
4	110　01011	Write	11	;　11 番地のデータを読み込んで OUTPUT REGISTER に出力・表示
5	000　00001	Jump	1	;　1 番地にジャンプする
6	011　01100	Load	12	;　12 番地のデータを読み込んで Accumulator に格納。
7	001　00000	Add	0	;　Accumulator に 0 番地(1)を加える。
8	100　01100	Store	12	;　12 番地に Accumulator の内容を格納
9	000　00001	Jump	1	;　1 番地にジャンプする
10	000　00000	Stop		;　終了
11	000　00000		0	;　表示するデータの保存場所
12	000　00000		0	;　10000000 から 1111111 までを表示するためのカウンタ

III　サンプルプログラム 7.3

i　まず、上のプログラムを実行させる前に、このプログラムがどのように動くのかを、自分が SSC になったつもりで検討しておいてください。

Accumulator の内容を 31 番地に保存し、0 を 30 番地から順に、29 番地、28 番地と格納していき、最終的に 7 番地に 0 を書き込んだ後 1 番地に戻り、6 番地に 0 を書きこんで、5 番地に Accumulator を書きこむ内容の命令を 1 番地に格納した後 6 番地で止まる。

ii　ACCUMULATOR の初期値が'0'でなかったら、結果はどうなるか考察してみましょう。

31 番地に Accumulator の初期値を書き込む以外は i で考察した動作と同じになる。（と、考えたが動かしてみたときの挙動がどうであったかあまり記憶にない。SSC は私の単純な推測通りに動いていなかった気がするのだが果たしてどうであったか。）

IV　課題 1

番地	機械語	ニモニック表記	コメント
0	000　00001	Jump　1	;　1 番地にジャンプする（無意味な命令）
1	110　01110	Write　14	;　14 番地の内容を読み込んで OUTPUT REGISTER に出力・表示
2	011　01110	Load　14	;　14 番地のデータを読み込んで Accumulator に格納
3	010　00000	Sub　0	;　Accumulator から 0 番地（1）を引く
4	100　01110	Store　14	;　14 番地に Accumulator の内容を格納
5	011　01111	Load　15	;　15 番地のデータを読み込んで Accumulator に格納
6	001　00000	Add　0	;　Accumulator に 0 番地（1）を加える。
7	100　01111	Store　15	;　Accumulator の内容を 15 番地に格納
8	000　00001	Jump　1	;　1 番地にジャンプする
9	011　10000	Load　16	;　17 番地のデータを読み込んで Accumulator に格納
10	001　00000	Add　0	;　Accumulator に 0 番地（1）を加える。
11	100　10000	Store　16	;　17 番地に Accumulator の内容を格納
12	000　00001	Jump　1	;　1 番地にジャンプする
13	000　00000	Stop	;　終了
14	111　11111	1111　1111	;　表示データの保存場所
15	000　00000	0	;　11111111 から 10000000 まで表示するためのカウンタ
16	000　00000	0	;　01111111 から 00000000 まで表示するためのカウンタ

1 つのカウンタを使ってループを回すだけで2^7回表示ができる。2 重のループでそれぞれ2^7回ずつ回すと’11111111’から’00000000’まで2^8種類の表示をすることが可能。つまり、1 番地から 8 番地をまわるループと 1 番地から 12 番地をまわるループが存在し、前者が’11111111’から’10000000’までを表示、後者が’01111111’から’00000000’までを表示する。

V　課題 2

番地	機械語	ニモニック表記	コメント
0	000　00001	Jump　1	;　1 番地にジャンプする（無意味な命令）
1	110　01110	Write　14	;　14 番地の内容を読み込んで OUTPUT REGISTER に出力・表示
2	011　01110	Load　14	;　14 番地のデータを読み込んで Accumulator に格納
3	111　00001	Shift　1	;　Accumulator の内容を 1bit 左シフトする
4	001　00000	Add　0	;　Accumulator に 0 番地(1)を加える。
5	100　01110	Store　14	;　Accumulator の内容を 14 番地に格納
6	000　00001	Jump　1	;　1 番地にジャンプする
7	110　01110	Write　14	;　14 番地の内容を読み込んで OUTPUT REGISTER に出力・表示
8	011　01110	Load　14	;　14 番地のデータを読み込んで Accumulator に格納
9	111　00001	Shift　1	;　Accumulator の内容を 1bit 左シフトする
10	100　01110	Store　14	;　Accumulator の内容を 14 番地に格納
11	000　00001	Jump　1	;　1 番地にジャンプする
12	011　00000	Load　0	;　0 番地のデータを読み込んで Accumulator に格納
13	000　00111	Jump　7	;　7 番地にジャンプする
14	000　00000	000　00000	;　表示データの保存場所

‘00000000’から’00000001’、’00000011’と順に表示する。これを、1bit シフトした後 1 を足すことで成立させた。’01111111’までの表示を 1 番地から 6 番地までのループで行っている。’11111111’を Accumulator に格納すると 6 番地の Jump 命令に引っかからず 7 番地の Write 命令で’11111111’を出力する。その後、’11111110’、’11111100’と表示していくことは 1bit シフト命令単体で実現している。7 番地から 13 番地のループで’11111111’から’10000000’の表示を行う。そして、7 番地に戻り、シフト命令で Accumulator に’00000000’を格納したとき、11 番地の Jump 命令に引っかかり 1 番地に戻って表示の無限ループが成立している。

VI　課題 3

番地	機械語	ニモニック表記	コメント
0	000　00001	Jump　1	;　1 番地にジャンプする(無意味な命令)
1	101　10100	Read　20	;　a の値をトグルスイッチから読み込んで 20 番地に格納
2	011　10100	Load　20	;　20 番地の内容を読み込んで Accumulator に格納
3	000　00110	Jump　6	;　6 番地にジャンプする
4	110　10011	Write　19	;　19 番地の内容を読み込んで OUTPUT REGISTER に出力・表示
5	000　00000	Stop	;　終了
6	011　10100	Load　20	;　20 番地の内容を読み込んで Accumulator に格納
7	010　00000	Sub　0	;　Accumulator から 0 番地の内容(1)を引く
8	000　01011	Jump　11	;　11 番地にジャンプする
9	110　10011	Write　19	;　19 番地の内容を読み込んで OUTPUT REGISTER に出力・表示
10	000　00000	Stop	;　終了
11	011　10011	Load　19	;　19 番地の内容を読み込んで Accumulator に格納
12	001　00000	Add　0	;　Accumulator に 0 番地の内容(1)を足す
13	100　10011	Store　19	;　Accumulator の内容を 19 番地に格納
14	011　10100	Load　20	;　20 番地の内容を読み込んで Accumulator に格納
15	010　10011	Sub　19	;　Accumulator から 19 番地の内容を引く
16	100　10100	Store　20	;　Accumulator の内容を 20 番地に格納
17	011　00000	Load　0	;　0 番地の内容を Accumulator に読み込む
18	000　00010	Jump　2	;　2 番地にジャンプする
19	000　00000	0	;　n の値の保存場所
20	000　00000	0	;　a の値の保存場所

Read 命令で読み込んだ a の値に対して、1 から n までの和が a より大きい、または一致した場合の n の値を出力する。その判別法は、a の値から 1 から順に引き算を行っていくことで判断する。n の初期値は 0 としてある。Load 命令で a の値を読み込み、3 番地の Jump 命令で正負を判定する。a の値が負であった場合は既に n までの和が a より大きいことを意味しているので n の値を出力して終了する。また、正だった場合、6 番地に飛んで 20 番地を読み込んで 1 を引く。そして Jump 命令で正負を判定する。ここで負になった場合、a は 0 であったということで一致の検出がされる。ここでも同じく 19 番地(n)の値を出力して終了する。正の場合は 11 番地に飛び、n に 1 を足して a から引き 20 番地に格納する。そして、2 番地に戻り判定を繰り返す。

VII 課題 4

番地	機械語	ニモニック表記	コメント
0	000 00001	Jump 1	; 1 番地にジャンプする(無意味な命令)
1	101 10000	Read 16	; トグルスイッチの内容を読み込んで 16 番地に格納
2	011 10000	Load 16	; 16 番地の内容を読み込んで Accumulator に格納
3	000 00110	Jump 6	; 6 番地にジャンプする
4	110 01111	Write 15	; 15 番地の内容を読み込んで OUTPUT REGISTER に出力・表示
5	000 00000	Stop	; 終了
6	011 10000	Load 16	; 16 番地の内容を読み込んで Accumulator に格納
7	010 00000	Sub 0	; Accumulator から 0 番地の内容(1)を引く
8	000 01011	Jump 11	; 11 番地にジャンプする
9	110 01110	Write 14	; 14 番地の内容を読み込んで OUTPUT REGISTER に出力・表示
10	000 00000	Stop	; 終了
11	110 01101	Write 13	; 13 番地の内容を読み込んで OUTPUT REGISTER に出力・表示
12	000 00000	Stop	; 終了
13	000 01111		; α>0 の時に表示する内容
14	111 11111		; α=0 の時に表示する内容
15	111 10000		; α<0 の時に表示する内容
16	??? ?????		αの値を保存する場所

「Accumulator のαについて・・・」と問題文にあったが、Accumulator を指定する方法が面倒であったので Read と Load で実現した。Load 命令で Accumulator にαの値を読み出す。そして、3 番地の Jump 命令でαの値が正か負かを判別する。負だった場合は 4 番地に進み'11110000'を表示して終了する。正だった場合は、6 番地に進み、20 番地の内容を読み込んで 1 を引く。そして、8 番地の Jump で正負を判定。符になるのはαが 0 だった場合であるのは 3 問目と同じ。'11111111'を出力・表示して終了する。正だった場合は 11 番地に飛んで'00001111'を出力して終了。

問題の内容からして 3 番目、5 番目を解くのに必要な基本的な考え方を聞いていると思った。順番としてはこれが 3 番目であるのが自然ではないか。

VIII課題 5

番地	機械語	ニモニック表記		コメント
0	000 00001	Jump	1	; 1 番地にジャンプする(無意味な命令)
1	101 11010	Read	26	; トグルスイッチの内容を読み込んで 26 番地に格納
2	011 11010	Load	26	; 26 番地の内容を読み込んで Accumulator に格納
3	000 01000	Jump	8	; 8 番地にジャンプする
4	011 11111	Load	31	; 31 番地の内容を読み込んで Accumulator に格納
5	000 10000	Jump	16	; 16 番地にジャンプする
6	011 00000	Load	0	; 0 番地の内容を読み込んで Accumulator に格納
7	000 01101	Jump	13	; 13 番地にジャンプする
8	011 11111	Load	31	; 31 番地の内容を読み込んで Accumulator に格納
9	000 01101	Jump	13	; 13 番地にジャンプする
10	110 10111	Write	23	; 23 番地の内容を読み込んで OUTPUT REGISTER に出力・表示
11	011 00000	Load	0	; 0 番地の内容を読み込んで Accumulator に格納
12	000 00001	Jump	1	; 1 番地にジャンプする
13	011 11010	Load	26	; 26 番地の内容を読み込んで Accumulator に格納
14	010 11111	Sub	31	; Accumulator から 31 番地の内容を引く
15	000 10011	Jump	19	; 19 番地にジャンプする
16	110 11001	Write	25	; 25 番地の内容を読み込んで OUTPUT REGISTER に出力・表示
17	011 00000	Load	0	; 0 番地の内容を読み込んで Accumulator に格納
18	000 00001	Jump	1	; 1 番地にジャンプする
19	010 00000	Sub	0	; Accumulator から 0 番地の内容(1)を引く
20	000 01010	Jump	10	; 10 番地にジャンプする
21	110 11000	Write	24	; 24 番地の内容を読み込んで OUTPUT REGISTER に出力・表示
22	000 00000	Stop		; 停止
23	000 01111			; $\gamma > \beta$ の時に表示する内容
24	111 11111			; $\gamma = \beta$ の時に表示する内容
25	111 10000			; $\gamma < \beta$ の時に表示する内容
26	000 00000			; β の値の保存場所
31	??? ?????			; γ の値の保存場所

4 問目の内容に Read を繰り返す要素を付け足した問題、Jump 命令による分岐が多くなっているのでその流れを手書きで図を用いて説明する。

IX　課題 6

番地	機械語	ニモニック表記	コメント
0	000　00001	Jump　1	;　1 番地にジャンプする（無意味な命令）
1	011　01110	Load　14	;　14 番地の内容を読み込んで Accumulator に格納
2	001　01111	Add　15	;　Accumulator に 15 番地を加える
3	100　01110	Store　14	;　Accumulator の内容を 14 番地に保存
4	011　01111	Load　15	;　15 番地の内容を読み込んで Accumulator に格納
5	001　00000	Add　0	;　Accumulator に 0 番地（1）を加える。
6	100　01111	Store　15	;　Accumulator の内容を 15 番地に保存
7	010　00000	Sub　0	;　Accumulator から 0 番地（1）を引く
8	010　10000	Sub　16	;　Accumulator から 16 番地の内容を引く
9	000　01100	Jump　12	;　12 番地にジャンプする
10	011　00000	Load　0	;　0 番地のデータを Accumulator に格納
11	000　00001	Jump　1	;　1 番地にジャンプする
12	110　01110	Write　14	;　14 番地の内容を読み込んで OUTPUT REGISTER に出力・表示
13	000　00000	Stop	;　終了
14	000　00000	0	;　求めた和の保存場所
15	000　00001	1	;　次に足す数の保存場所
16	000　01010	10	;　足す数の上限（10）を保存する場所

Σ記号による 1 から n までの整数の和を求める問題。一応 16 番地に収める n の値によって判断するようにしているがあまり工夫はしていない。n が 1 以上であることを想定している。

1 番地で和を保存する 14 番地を読み込み、15 番地に入れてある次に足す数を足す。14 番地に書き出して保存。次に、15 番地の値を読み込み、1 を足して保存。ここで 1 を引いて、さらに 16 番地の内容を引いて、これが正になる時に加算のループから脱出する。具体的には 15 番地が 8 だったときは次に足す数は 9 になる。ここから 1 引いて 8。10 を引いて-2。よって、そのまま 1 番地に戻って処理を続行する。10 を足した後に 15 番地は 11 に変わり、1 を引いて 10 を引くと 0。ここで Jump 命令に引っかかりループを脱出する。16 番地に 10 を入れるための工夫であり、x=0 の判定で用いた方法と同じでもある。

X　調査事項 現在、ノイマン型以外のどんな形式の計算機システムが考えられているか

量子計算機(1bit に 0, 1 のだけを使う 2 進数ではなく、0 と 1 のあいだの数字を任意の重ねあわせで保持する計算機)

DNA コンピューター(DNA の塩基結合を利用した計算機。ATGC の塩基と DNA を操作する酵素によって計算を行う。計算操作は早いが結果を分析するのに膨大な時間がかかる。)

ニューロコンピュータ(脳の構造(神経細胞のネットワーク構造)を応用した計算機。一瞬で最適な方法を選択するなどの人間(生物)が得意な行動を、計算機が虱潰しに計算するときに応用する)

XI　感想・意見

私が 2 年ということと、計算機通論で MIPS アセンブリの知識を得ていたこともあり、講義の内容と課題自体に特に疑問をもつこともなかったし、分からないということもなかった。

しかし、学部の 1 年の知識量を考えると少し気にかかることがあった。1 年次にコンピューターリテラシーでやることは東 3 号棟の演習室の利用法とネチケット程度で、人によってはプログラミングの知識が皆無であることもある。実際、去年の後期の基礎プログラミング演習でも、C 言語を扱った際、単純な手続き型言語の操作に四苦八苦しているのが大半であった。最悪、授業外で所謂分かっている学生が他の学生に対して講習会まがいのことを行っている状態でもあった。と、事実上前期のこの時期ではプログラミングの知識ゼロで Slow Scan Computer と言われても……という人も多かったのではないか。課題として出されたものの中には 5 番の様にちょっと考えないと分からないような類の問題もあった。確かに、解き終わらないことが前提という先生の話は的をいているのだろうが、プログラムの作成課題については、例題を出して考える時間、解答と解説という時間がもっと明確に存在してもよかったかと思う。

Slow Scan Computer と言っても、本当に Slow だった。課題の 1 や 7.2 の様に2^8回ループを回そうとすると簡単に 30 分、40 分という時間が経ってしまう。早く回す手段が欲しい。むしろ、確認用に 2 台目の使用を行ってもよいかどうかを聞いてみればよかったと今頃ながら思った。

このテーマ 3 のテキスト片手にプログラムを考えていると、サークルの先輩と話が弾んだのが面白かった。専門実験で扱ったようだが内容としてはある程度似てくるのだろうか。トグルスイッチで入れていくのが面倒で…などと非常に盛り上がった。とはいえ、こういう単純な構造のコンピューターに触れるのはやはり楽しい時間であったし、工房という場だからこと分野を変えて色々と掻い摘んでやれるのは楽しい。