1. アーキテクチャ

(A) 入出力端子

信号	入出力	用途	説明
clock	入力	クロック	エッジトリガ(立ち上がり)　CPU の状態が遷移する
reset	入力	リセット	エッジトリガ(立ち下がり)　クロックに優先して CPU をリセットする
iport[7:0]	入力	入力ポート	CPU に値を入力する　任意のタイミングで変化してよい CPU はクロックが立ち上がるとき、iport の値を保持する
oport[7:0]	出力	出力ポート	CPU から値を出力する CPU はクロックが立ち上がるとき、oport の値が変化する

(B) レジスタ

レジスタ	用途	説明
pc[7:0]	プログラムカウンタ	
acc[7:0]	アキュムレータ	
flag[1:0]	フラグレジスタ flag = { C , Z }	C:acc でキャリーが発生するとき 1 Z: acc が 0 になるとき 1
ix[7:0]	インデックスレジスタ	間接アドレッシングモードで使用
iy[7:0]	インデックスレジスタ	間接アドレッシングモードで使用
iport	入力レジスタ	レジスタ iport は、クロックが立ち上がるとき、入力端子 iport の状態を保持する
oport	出力レジスタ	レジスタ oport は、クロックが立ち上がるとき、出力端子 oport に値を出力する

(C) メモリ

メモリ	分類	説明	容量
IM (instruction memory)	ROM	プログラムを格納する	24bit × 16 (最大 256)
DM (data memory)	RAM	データを格納する	8bit × 固定 256

(D) 特別なデータアドレス

外部入出力端子 iport,oport は memory maped IO 方式でメモリ上に配置する。

ix(iy)レジスタはメモリ上に配置する。ix(iy)レジスタ用の特別な命令セットは用意しない。

アドレス	名称	用途	
F0	oport	8 ビット出力ポート	
F1	iport	8 ビット入力ポート	
F2	ix	ix レジスタ	
F3	iy	iy レジスタ	
F4-FF	なし	後の拡張のため予約	

2. 命令セット

命令	マシン語	機能	値が変化するレジスタ	
ADD	0000	加算	acc,Z,C	
ADC	0001	キャリーを含めて加算		
SUB	0010	減算		
SUC	0011	キャリーを含めて減算		
AND	0100	論理積		
OR	0101	論理和		
XOR	0110	排他的論理和		
NOT	0111	論理否定		
LD	1000	acc やメモリに値を入れる		
B	1011	条件分岐	なし	
FIN	1111	シミュレーション終了	なし	

3. アドレシングモード

(A) ADD,ADC,SUB,AND,OR,XOR,NOT,LD 命令

ADD,ADC,SUB,AND,OR,XOR,NOT,LD の各命令は 2 つのオペランドをもつ。
oprand1 は#m,@m,$m,acc のいずれかが使用できる。oprand2 は@m,$m,acc のいずれかが使用できる。

モード	記述	説明	使い方	処理
イミディエイト(即値)	#m	オペランドを値とする	LD #5 @3	値 5 をデータメモリ 3 番地に格納
直接アドレシング	@m	オペランドをアドレスとする	LD acc @3	アキュムレータ acc の内容をデータメモリ 3 番地に格納
間接アドレシング	$m	ix(iy)レジスタ+オペランドをアドレスとする	LD $3 $4 (ix=5,iy=2)	8 番地(ix レジスタ+3)の内容を 6 番地(iy レジスタ+2)に格納
アキュムレータ	acc	アキュムレータを指定する	LD #0 acc	値 0 をアキュムレータ acc に格納

(B) B 命令

B 命令(条件分岐命令)では相対アドレスのみ使用できる。B 命令は 2 つのオペランドをもつ。
oprand1 はフラグ(Z,C,A)が使用できる。oprand2 は相対アドレス値 m を指定する。

モード	記述	使い方	処理
相対アドレス	m	B A 251	常に pc-5 番地から実行する
		B C 3	C フラグが 1 であるとき pc+3 番地から実行する C フラグが 0 であるときは pc+1 番地から実行する
		B Z 251	Z フラグが 1 であるとき pc-5 番地から実行する Z フラグが 0 であるときは pc+1 番地から実行する
絶対アドレス	----	----	サポートしない

4. プログラム例

(A) 1 から iport までの和を計算する

iport から値 n を読み込み、 $\sum_{i=1}^{n} i$ を計算する

address	label	op	oprand1	oprand2			comment
00		LD	#0	@sum			0→@sum
01		LD	@iport	@n			@iport→@n
02	LOOP:	ADD	@sum	@sum			acc+@sum→@sum
03		LD	@n	acc			@n→acc
04		SUB	#1	@n			acc-1→@n
05		B	Z	EXIT			if Zflag EXIT
06		B	A	LOOP			else LOOP
07	EXIT:	LD	@sum	@oport			@sum→@oport
08		LD	acc	acc			なにもしない
09		FIN					$finish

(B) 課題検討用

リセット後、フィボナッチ数列(1,1,2,3,5,...)を oport に出力する。フィボナッチ数列が 255 を超えるとき(C フラクがが 1)はシミュレーションを終了する。

address	label	op	oprand1	oprand2			comment
00							
01							
02							
03							
04							
05							
06							
07							
08							
09							
0A							
0B							
0C							
0D							
0E							
0F							

5. ハンドアセンブル
ニーモニックをマシン語に変換する作業を「アセンブル」いい、この変換作業を代行するソフトウェアを「アセンブラ」という。この CPU にはアセンブラは存在しないので手作業でアセンブルする。このことを「ハンドアセンブル」という。

(A) 準備
ソースプログラム全体を見て、ラベルと変数を把握する。
変数(@xxx)にはデータメモリ DM 上のアドレスを割り当てる。@oport,@iport は CPU 上で指定された値を使用する。@sum,@n は適当なアドレスでよい。
ラベルはインストラクションメモリ IM 上のアドレスを読み取る。この CPU ではソースプログラムの行番号を IM 上のアドレスとして扱うことができる。

ラベル	IM 上のアドレス
LOOP:	02
EXIT:	07

変数	DM 上のアドレス	
	16 進表示	2 進表示
@oport	F0　指定値	1111 0000
@iport	F1　指定値	1111 0001
@sum	04	0000 0100
@n	05	0000 0101

(B) LD,ADD,ADC,SUB,AND,OR,XOR,NOT 命令
マシン語各命令は 24 ビット固定長であり、その構成は図のようになっている。

命令(4bit)	アドレシングモード(4bit)		source (8bit)	destination (8bit)
	source (2bit)	destination (2bit)		

命令セット、アドレシングとマシン語(2 進表示)の対応を示す。

命令	
ADD	0000
ADC	0001
SUB	0010
SUC	0011
AND	0100
OR	0101
XOR	0110
NOT	0111
LD	1000

アドレシングモード	
#	00
acc	01
@	10
$	11

命令セットを参考に命令(op)を 2 進数に変換する。アドレシングモードを参考に#,@,$,acc を 2 進数に変換する。イミディエイトモードは値をそのまま 2 進数にする。変数(@xxx)は先の表を参考にアドレスを 2 進数にする。
(1)から(5)の順に並べ直すとマシン語になる。後の作業のため、16 進表示も記入する

op	oprand1		oprand2			HEX	マシン語
LD	#	0	@	sum			
1000	00	0000 0000	10	0000 0100		820004	1000 0010 00000000 00000100
(1)	(2)	(4)	(3)	(5)			(1)　(2)(3)　(4)　　(5)
LD	@	iport	@	n			
1000	10	1111 0001	10	0000 0101		8AF105	1000 1010 11110001 00000101
ADD	@	sum	@	sum			
LD	@	n	acc				
SUB	#	1	@	n			
LD	@	sum	@	oport			
LD	acc		acc				

address	label	op	oprand1	oprand2	HEX	マシン語
00		LD	#0	@sum	820004	1000 0010 00000000 00000100
01		LD	@iport	@n	8AF105	1000 1010 11110001 00000101
02	LOOP:	ADD	@sum	@sum		
03		LD	@n	acc		
04		SUB	#1	@n		
05		B	Z	EXIT	後述	
06		B	A	LOOP	後述	
07	EXIT:	LD	@sum	@oport		
08		LD	acc	acc		
09		FIN			後述	

(C) B 命令

分岐条件は A,C,Z のいずれかを指定する。A,C,Z は 4 ビットの値に変換される。
分岐先アドレスと現在の pc から相対アドレス値 m を計算する。

命令　(4bit)	フラグ　(4bit)	相対アドレス値 m　(8bit)
1011	A : 0000 C : 1000 Z : 0100	(相対アドレス値 m) = (分岐先アドレス) ー (現在の pc)

2	LOOP:			LOOP:　02 番地
3				
4				
5	B　Z　EXIT			
6	B　A　LOOP			
7	EXIT:			EXIT:　07 番地

EXIT は 07、現在の pc は 05　相対アドレス値 m は 7-5=2 (0000 0010)

5	B　Z　EXIT			
		固定値		
	1011	0100	0000 0000	0000 0010

LOOP は 02、現在の pc は 06　相対アドレス値 m は 2-6=-4

6	B　A　LOOP			
		固定値		
	1011	0000	0000 0000	1111 1100　0000 0100 の補数

コーディングシートに 16 進表示と 2 進表示を記入する。

5	B　Z　EXIT		B40002	1011 0100 0000 0000 0000 0010	
6	B　A　LOOP		B000FC	1011 0000 0000 0000 1111 1100	

(D) FIN 命令

シミュレーションを終了する。

命令　(4bit)	固定値　(4bit)	固定値　(16bit)
1111	0000	0000 0000 0000 0000

(E) 完成

以上よりハンドアセンブルは完了した。
マシン語(2 進表示)はマシン語実行ファイルとして保存する。16 進表示は、デバッグ・動作確認時に使用する。(空欄は各自で埋めること)

address	label	op	oprand1	oprand2	HEX	マシン語(2 進表示)
00		LD	#0	@sum	820004	1000 0010 00000000 00000100
01		LD	@iport	@n	8AF105	1000 1010 11110001 00000101
02	LOOP:	ADD	@sum	@sum		
03		LD	@n	acc		
04		SUB	#1	@n		
05		B	Z	EXIT	B40002	1011 0100 0000 0000 0000 0010
06		B	A	LOOP	B000FC	1011 0000 0000 0000 1111 1100
07	EXIT:	LD	@sum	@oport		
08		LD	acc	acc		
09		FIN			F00000	1111 0000 00000000 00000000

6. マシン語実行ファイル

ここまでの作業で完成したマシン語を実行ファイルとして保存する。ファイル名は「sample.hex」とする。

820004 8AF105 ・・・ F00000 0 0 ・・・ 0	作成したプログラムのあとに 0 を追加し、全体で 16 行にする ファイル名「sample.hex」で保存する

7. 回路記述・テストベンチ

Verilog HDL により CPU の回路記述およびテストベンチを作成する。

8. シミュレート

端末からコマンドを入力し、シミュレートする。

iverilog -o tmp testbench.v cpu.v …他必要なファイル vvp tmp gtkwave tmp.vcd	下位モジュールを含むファイルを忘れずに

シミュレーションではタイミングチャートでつぎの各項目を確認する。

□sample.hex が正しく実行できているか。
□プログラム(マシン語)は正しい順番に ir レジスタに格納されたか。
□計算過程において、acc,@sum,@i は正しいタイミングで変化しているか。
□pc が変化するタイミングと Instruction Fetch のタイミングは正しいか。
□各命令・アドレシングモードについて Instruction Decode 時に各レジスタは正しく変化したか。
□Execute 時にフラグと acc は正しく変化したか。
□分岐命令において nextpc,pc が変化するタイミングは正しいか。

9. 付録　コーディングシート

address	label	op	oprand1	oprand2	HEX	BIN	comment
00							
01							
02							
03							
04							
05							
06							
07							
08							
09							
0A							
0B							
0C							
0D							
0E							
0F							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
1A							
1B							
1C							
1D							
1E							
1F							
20							