情報工学実験 (ハードウェア実験) 報告書

PARTHENONによる32ビットマイクロプロセッサの作成

学生番号: 09425566
提出者: 佐藤 佑太

提出日: 2015年 7月 27日
締切日: 2015年 7月 27日

概 要

本稿では，情報工学実験（ハードウェア実験）において作成した、32 ビットマイクロプロセッサについてまとめる。

1 はじめに

本実験の目的は、(1) ハードウェア記述言語と CAD ツールを使えるようになる、(2) マイクロプロセッサを設計することによって、論理回路やコンピュータアーキテクチャについてどのように動いているのかの根本を理解する である。

本報告書では，ハードウェア記述言語と CAD ツールを用いた論理回路の設計と、32 ビットマイクロプロセッサの設計について報告する．

本報告書の構成は次のとおりである．まず **2** にて本実験で設計したプロセッサの概要について述べる．

3 にて本実験における実施内容の状況報告を行う。

4 では，アセンブリによって作成したプログラムの作成に関しての報告を行う。

5 では，作成したプロセッサの設計に関して報告を行う。

6 では，発展課題で取り組んだ課題について報告する。

7 では，諸事項について検討を行い、それについての考察を記述する。

8 では，設計の際に工夫した点や特に注力した点について報告する。

9 では，本実験を通して、どこれまでから一層理解が深まった部分などについて記述する。

10 では，本実験の進捗状況報告ファイルを掲載する。

11 では，本実験で作成したアセンブリ言語プログラム、設計した SFL 記述、テスト用スクリプト、テスト結果、論理合成時の出力等のファイルの一覧を掲載する。

最後に **12** で，本報告のまとめと今後の課題を述べる。

2　設計したプロセッサの概要

ここでは，設計したプロセッサの概要について述べる．

本実験で設計したプロセッサは 32 ビット RISC マイクロプロセッサとなっており、コプロセッサやキャッシュメモリは実装されていない。

アセンブリ言語は MIPS のものとほぼ同様のものとなっている。

また、本実験では 3 種類のプロセッサを実装する。各プロセッサの構造は以下のようになっている。

1. p32m1
   (a) 1 命令を 5 サイクルで実行する
   (b) パイプライン実行は無し
2. p32m2
   (a) 1 命令を 2 5 サイクルで実行する
   (b) パイプライン実行は無し
3. p32p1
   (a) 1 命令を 5 ステージパイプラインで実行する

p32m1, p32m2 のプロセッサは、1 つの命令に対して 5 サイクル使う。つまり、1 つの命令が終了するまで次の命令を実行せず、その命令が終了してから次の命令を実行するような方式になっている。

対して p32p1 はパイプライン実行を考慮した設計を行う。今回設計するプロセッサは 5 つのステージを持つため、同時で最大 5 命令実行することになる。

ただし、パイプライン実行処理を行う場合は、前後の命令が使うレジスタの値などを考慮する必要があり、データフォアワーディングという処理を行う必要がある。

データフォアワーディングとは、前の命令が後の命令がレジスタに書き込む値を利用しなければならない場合などに、MEM ステージなどでロードされた値を、前の命令が EX ステージで利用できるように渡してやる処理のことである。

これを行うことによって、パイプライン実行時に各命令が適切なレジスタの値を利用することができる。

本プロセッサで使用するサブモジュールは、以下である。

- レジスタファイル
  - 32 ビット x32 本の汎用レジスタ
  - 入力は 1 ポート
  - 出力は 2 ポート
- 32 ビット ALU

  - 32 ビットの算術・論理演算

- 32 ビットシフタ
  - 論理シフト・算術シフト

なお、今回は乗算器、徐算器は用いない。
今回の 5 ステージパイプライン構造は、以下のステージにより構成されている。

- IF: 命令メモリから命令をフェッチする
- ID: レジスタの値を取り出したり、分岐処理を行ったりする
- EX: 演算を実行する
- MEM: メモリアクセス関連の処理を行う
- WB: レジスタに書き込む

2.1 サポートする命令セット

まず、本実験でサポートする命令セットを以下の表に示す。

表 1: サポートする命令一覧

命令種別	サポートする命令	サポートしない命令
算術論理演算	add, addu, addi, addiu, sub, subu, and, andi, or, ori, xor, xori, nor	mult, multu, div, divu
比較	slt, sltu, slti, sltiu	
シフト	sll, srl, sra, sllv, srlv, srav	
ロートストア	lw, sw, lb, sb	
分岐	beq, bne	
ジャンプ	j, jr, jal, jalr	
データ転送	lui, mfhi, mflo	
例外，システムコール	syscall	

今回は掛け算、割り算の命令については時間が足りなかったため実装を試みることができなかった。

3 実施状況の報告

ここでは，取り組んだ課題の実施状況について報告する。

3.1 報告内容に関する事項

私たちの班では、役割分担をせずにそれぞれがプログラムの作成を行った。そのため、各プログラムにおける個人の役割は 100 中全員がすべて 100 であると考えたため、表からは省略している。

課題の実施内容は、以下の表のようになっている。

表 2: 実施状況

設計課題番号	内容	実施
D-1	32 ビット加算器 add32 の設計	実施済み
D-2-1	カウンタの設計	実施済み
D-2-2	カウンタの設計	未実施
D-2-3	カウンタの設計	未実施
D-3	32 ビット ALU の設計	実施済み
D-4	32 ビットシフタの設計	実施済み
E-1	32 ビット Carry Lookahead Adder の設計	未実施
E-2	32 ビット整数乗算器の設計	未実施
E-3	32 ビット除算器の設計	未実施
5-1	5 ビット比較器	実施済み
5-2	レジスタファイル	実施済み
5-3	メモリユニット	実施済み
6-1	p32 プロセッサコア (マルチサイクル)	実施済み
E6-1	p32 プロセッサコアの改良	未実施
E6-2	乗算機能の実装	未実施
7-1	p32 プロセッサコア (マルチサイクル v2)	実施済み
E7-1	p32 プロセッサコアの改良	未実施
E7-2	乗算機能の実装 (2)	未実施
8-1	p32 プロセッサコア (パイプライン)	実施済み
E8-1	p32 プロセッサコアの改良	未実施

4　プログラミング課題に関する報告

ここでは，アセンブリで実装したプログラミング課題 3 5 に関しての報告を行う。

4.1　プログラミング課題 3

プログラミング課題 3 では、整数をソートするプログラムの作成を行った。

与えられた n 個の 32 ビットの整数データを降順にソートするプログラムであり、私はバブルソートのアルゴリズムを用いてソートを行うプログラムを作成した。

バブルソートでは、全ての要素に対して隣り合う要素と数値の大小を比較し順序が逆であれば入れ替える。このためオーダーは O(n$\hat{2}$) となり性能はよくならない。

余裕があればクイックソートでもソートプログラムを作成してみたいと考えていたが、実装のしやすさからバブルソートによるソートプログラムの作成を行った。

4.2　プログラミング課題 4,5

今回は実装することができなかった。

5　プロセッサ設計課題に関する報告

ここでは，プロセッサ設計課題に関する報告を行う。

5.1　設計課題 6-1

設計課題 6-1 では、マイクロプロセッサ p32 のコアモジュールとなる p32m1,p32m2,p32p1 の設計を行った。

各サブモジュールを論理合成した結果、以下の表のような数値が得られた。

表 3: 論理合成で得られた諸量のまとめ

モジュール	最大遅延 (ns)	最大動作周波数 (MHz)	ゲート数	実装面積 ($1000\mu m^2$)	消費電力 (μW/MHz)	最大動作周波数で動作時の消費電力
32 ビット加算器	33.1	30.21	95	25.18	224.0	6.77
32 ビット ALU	125.3	30.21	95	25.18	224.0	6.77
32 ビットシフタ	26.6	27.32	2488	267.91	2519.8	68.85
5 ビット比較器	24.0	41.67	27	6.26	67.1	2.80
レジスタファイル	31.6	31.65	17871	4888.41	34645.4	1096.37
p32 プロセッサコア (マルチサイクル)	125.0	8.00	30512	8240.88	57859.9	462.88
p32 プロセッサコア (マルチサイクル v2)	124.5	8.03	30994	8357.85	58670.4	471.25
p32 プロセッサコア (パイプライン)	230.9	4.33	31991	8597.80	60738.8	263.05

6　追加課題や発展課題に関する報告

今回は追加課題や発展課題に取り組む時間を確保することができず、実装することができなかった。

7　検討・考察

7.1　プログラミング課題 3 の実行結果

以下に、プログラミング課題 3 で作成したプログラム、p32m1,p32m2,p32p1 で実行した場合の実行性能をまとめた表を示す。

表 4: プログラミング課題 3 の実行結果

コアプロセッサ (データ数)	実行時間	実行エネルギー	エネルギー時間積
p32m1(32)	0.00419125	0.00194005	0.00008980084
p32m1(128)	0.06162	0.02852267	0.0132025715
p32m1(512)	0.97679826	0.452140377	0.2093867377
p32m2(32)	0.0031855056	0.00150096	0.0007073263
p32m2(128)	0.0469599	0.02212985	0.0104286933
p32m2(512)	1.38781155	0.65400619	0.3082004187
p32p1(32)	0.00154956	0.00040764	0.0001072287
p32p1(128)	0.0123245	0.00324196	0.0008527975
p32p1(512)	0.36229607	0.09530198	0.0250691864

どのデータ数に対しても、p32p1 が一番速い処理を終了していることが表からわかる。これはパイプライン処理を行っていることが要因になっていると分かる。

また全体的にプロセッサ自体を仕上げることのみにしか時間を割くことができず、細部でよりよいパフォーマンスをだすための工夫などを施すことができなかったのが残念であった。

8　工夫した点や特に力を注いだ点

プロセッサに使われている各モジュール仕組みを逐一しっかりと理解しながらプロセッサを設計することをこころがけた。そうすることでよりプロセッサの挙動を理解しながら設計し、そこから性能をあげるような実装もできるのではないかと考えた。

ただ今回の実験中にはそれらを実践していろいろな工夫を施すことにまだまだ余地があったなと残念に思う。

9　本実験を実施して得られたこと

本実験を通して、コンピュータの核であるプロセッサがどのようなモジュールで成り立ち、その役割を果たしているのかの基礎を学ぶことができた。

特に C 言語などの高級言語の中で比較的低級なプログラムを作成している時には、内部でプロセッサがどのように演算を行ってプログラムで求めている答えを導出するのか意識しながらプログラムを書けることも多くなったように思う。

例えば計算機において和と差の計算は効率がよいが、積と商の計算が効率がわるくなるといったことをプロセッサの内部モジュールの観点から考えれるようになるということはエンジニアとして非常に大切なことのように感じる。

また、設計を通して、今回理解できたプロセッサの機能は非常に小さく初歩的なものであり、現在のコンピュータへの性能の期待を満たすパフォーマンスを持つプロセッサを作るには、キャッシュや並列・分散処理における高速化などさらに知るべきことがたくさんあることも同時に学ぶことができた。

今回学んだことは現在主流になっている、またはこれから主流になるような技術を理解する上で非常に役にたつものだと思うので、今回学んだことを活かしながらこれからの技術に対する理解もしていきたい。

10 進捗状況報告

この章では、設計課題の進捗状況を報告する。

設計内容の進捗は以下に掲載する。

（設計課題進捗状況）

グループ: B-10
学生番号: 09425566
氏　　名: 佐藤佑太
最終更新: 2015-06-23(火) 18:00

```
       | (2)  | | (3)  | | (4)  | | (5)  | | (6)  | | (7)  | | (8)  | | (9)  |
       | 5/26 | | 6/2  | | 6/9  | | 6/16 | | 6/23 | | 6/30 | | 7/7  | | 7/14 |
課題   | 3 4 5| | 3 4 5| | 3 4 5| | 3 4 5| | 3 4 5| | 3 4 5| | 3 4 5| | 3 4 5|
-------+------+-+------+-+------+-+------+-+------+-+------+-+------+-+------+
(D-1)  |  <+T                                                              S>
(D-2-1)|    <+ttttttttttttT>
(D-2-2)|
(D-2-3)|
(D-3)  |                     <+tttT                                        S>
(D-4)  |                            <++ttT                                 S>
(E-1)  |
(E-2)  |
(E-2)  |
```

```
(5-1)  |                                  <+T                                  S>
(5-2)  |                                     <+T                               S>
(5-3)  |                                        <++T>
(6-1)  |                                             <++++ttttttttttT         S>
(E6-1) |
(E6-2) |
(7-1)  |                                                   <+++++tttttttT     S>
(E7-1) |
(E7-2) |
(8-1)  |                                                                <+tttT S>
(E8-1) |
-------+------+-+------+-+------+-+------+-+------+-+------+-+------+-+------+
```

（シンボルとその説明）

- < ... 着手
- + ... 実施（コーディング）
- t ... Seconds シミュレーション，デバッグ
- T ... Seconds シミュレーション，デバッグ 完了
- v ... Verilog HDL シミュレーション，デバッグ
- V ... Verilog HDL シミュレーション，デバッグ 完了
- S ... 論理合成
- > ... 完成
- * ... 実施（改良等）
- . ... その他

※ シンボルは半角文字で書くこと（全角文字は使わない）

（基本モジュールの設計）

(D-1) 【設計課題 2-1】〜【設計課題 4-1】 32 ビット加算器 add32 の設計
(D-2-1)【設計課題 2-2】〜【設計課題 4-2】 カウンタの設計 (_-2-1)
(D-2-2)【設計課題 2-2】〜【設計課題 4-2】 カウンタの設計 (_-2-2)
(D-2-3)【設計課題 2-2】〜【設計課題 4-2】 カウンタの設計 (_-2-3)
(D-3) 【設計課題 2-3】〜【設計課題 4-3】 32 ビット ALU の設計
(D-4) 【設計課題 2-4】〜【設計課題 4-5】 32 ビットシフタの設計
(E-1) 【発展課題 2-1】〜【発展課題 4-1】 32 ビット Carry Lookahead Adder の設計
(E-2) 【発展課題 2-2】〜【発展課題 4-2】 32 ビット整数乗算器の設計
(E-3) 【発展課題 2-3】〜【発展課題 4-3】 32 ビット整数除算器の設計
(5-1) 【設計課題 5-1】 5 ビット比較器
(5-2) 【設計課題 5-2】 レジスタファイル
(5-3) 【設計課題 5-3】 メモリユニット

（プロセッサの設計）
(6-1) 【設計課題 6-1】 p32 プロセッサコア（マルチサイクル）
(E6-1) 【発展課題 6-1】 p32 プロセッサコアの改良
(E6-2) 【発展課題 6-2】 乗算機能の実装
(7-1) 【設計課題 7-1】 p32 プロセッサコア（マルチサイクル v2）
(E7-1) 【発展課題 7-1】 p32 プロセッサコアの改良
(E7-2) 【発展課題 7-2】 乗算機能の実装 (2)
(8-1) 【設計課題 8-1】 p32 プロセッサコア（パイプライン）
(E8-1) 【発展課題 8-1】 p32 プロセッサコアの改良

11 作成した設計記述、プログラム等のリポジトリ名、ファイル名の一覧

本レポート末尾の資料 A に、作成した設計記述、プログラム等のリポジトリ名、ファイル名の一覧をまとめた表を掲載する。

なお、本実験を通して作成したプログラムはいずれも git でのヴァージョン管理を行っており、本実験用のホスト先 URL にあげている。URL は http://jikken1.arc.cs.okayama-u.ac.jp/gitbucket/09425566/processor である。

12 おわりに

本報告書では，本実験で設計したマイクロプロセッサについて報告した．本実験テーマの目的である (1) ハードウェア記述言語と CAD ツールを使えるようになる、(2) マイクロプロセッサを設計することによって、論理回路やコンピュータアーキテクチャについてどのように動いているのかの根本を理解するは，をそれぞれ達成することができた。設計の段階からプロセッサを構築することにより、コンピュータが根本ではどのようなモジュールが結合しあって動いているのかということと同時に、各モジュールがどのような役割をもちどのような実装をされることでその役割を果たす機能を提供するのか、ということがわかった。

今回はプログラムの改良に時間を割くことができなかったので、今後の課題としてどんな工夫や改善を行うことで、よりよい性能をだすことができるのか、ということについてももっと検討し、理解を深めていきたい。

Apple Computer を作ったスティーヴ・ウォズニアックは Apple I と Apple II を独力で作り上げ、そのときの様子が彼の伝記で描かれている。

本の中では内部の電子回路をいかにシンプルに設計するかという工夫のプロセスやフロッピーディスクの読み取り装置の設計など技術的に説明されているものも多く、この授業を受けたからこそ、そのすごみが今となって理解できるように思う。

コンピュータの歴史が抽象化の歴史であるならば、コンピュータサイエンスはその抽象化をひとつひとつ具体化していくような作業のように思う。

表層的にはシンプルに見える、うまく抽象化されたその技術の根本を知ることは技術者になる上で非常に大切なことだとこの実験を通して学んだ。

付録

A　本実験で作成したプログラムの一覧

全てのモジュールは本実験で利用した gitbucket のリポジトリで管理を行っている。リポジトリの URL は http://jikken1.arc.cs.okayama-u.ac.jp/gitbucket/09425566/processor である。

また、いかに本実験で作成したプログラムの一覧表を記載する。プログラムの場所は、上記 gitbucket のリポジトリ URL 以下のものとする。

表 5: 作成したプログラムの一覧

モジュール	フォルダ	SFL ファイル	テスト用スクリプト	テスト結果	論理合成出力ファイル
32 ビット加算器	/add32	add32.sfl	add32.sec	add32.result	logical_mix.result
32 ビット ALU	/alu32	alu32.sfl	alu32.sec	alu32.result	logical_mix.result
32 ビットシフタ	/shift32	shift32.sfl	shift32.sec	shift32.result	logical_mix.result
5 ビット比較器	/comp5	comp5.sfl	comp5.sec	comp5.result	logical_mix.result
レジスタファイル	/reg32x32	reg32x32.sfl	reg32x32.sec	reg32x32.result	logical_mix.result
メモリユニット	/memunit	memunit.sfl	memunit.sec	memunit.result	
p32 プロセッサコア (マルチサイクル)	/p32m1	p32m1.sfl	p32m1.sec	p32m1.result	logical_mix.result
p32 プロセッサコア (マルチサイクル v2)	/p32m2	p32m2.sfl	p32m2.sec	p32m2.result	logical_mix.result
p32 プロセッサコア (パイプライン)	/p32p1	p32p1.sfl	p32p1.sec	p32p1.result	logical_mix.result