平成 27 年度情報工学実験第一・第二のための春休み課題（その 1）

渡邊誠也 † 笹倉万里子 †† 佐藤将也 †††

この課題は，平成 27 年度情報工学実験第一・第二の準備として，実験受講予定者のみなさんに春休み中に解答していただくものです．この課題の解答を報告書にまとめて平成 27 年度の第 1 回目の実験の時間に持参してください．報告書には各課題の解答を記述してください．解答に記載するプログラムコード，フローチャートは解答箇所が分かるようにしてください．報告書は **A4 用紙**とし用紙の上側に学生番号と氏名を記載してください（表紙は不要）．手書きでも構いません．

1. （課題 1）C 言語プログラミング

1.1 問　　題

配列データの各要素の値の総和を計算する関数（手続き）vsum を C 言語で作成せよ．**図 1** に関数 vsum の C 言語プログラムコード（未完成版）を示す．図 1 の空欄部分を埋めてプログラムを完成させよ．

配列の各要素は int 型とし，アドレス *addr* から連続する *n* 個の要素の総和を計算する．計算結果は，関数の返値として呼び出し元に返すものとする．

1.2 説　　明

図 1 に示す関数 vsum の C 言語プログラムコードについて説明する．

int へのポインタ型の仮引数 `addr` に連続するデータの先頭アドレス *addr* が，int 型の仮引数 `n` に総和を求めるデータの個数 *n* がそれぞれ呼び出し元から渡される（2 行目）．3 行目で宣言および初期化されている int 型の変数 `sum` は，総和計算をする際に計算の途中結果を格納しておくために用いる．最終的にはこの変数 `sum` に求める総和が格納されることになる．4 行目で宣言されている int 型の変数 `i` は繰り返し文（for 文）で用いるループカウンタである．

6〜8 行目の繰り返し文（for 文）で総和計算を行う．9 行目で変数 `sum` に格納されている計算結果を呼び出し元に返す．

2. （課題 2）アセンブリ言語への変換

2.1 問　　題

課題 1 で作成した C 言語プログラムを MIPS アセンブリ言語[1] に変換せよ．配列の先頭アドレスはレジスタ`$a0` に，加算するワード数はレジスタ`$a1` にそれぞれ格納されているものとし，計算結果はレジスタ`$v0` に格納するものとする．また，手続きからの戻りアドレスはレジスタ`$ra` に格納されているものとする．

```
1   int
2   vsum(int *addr, int n) {
3       int sum = 0;
4       int i;
5
6       for (i = 0; i < n; i++) {
7           ------(1)------
8       }
9       return sum;
10  }
```

図 1 関数 vsum の記述 (vsum.c)

2.2 説　　明

2.2.1 プログラム変換

まず，プログラムの処理の流れを把握し，アセンブリ言語への変換に適した記述への変換を行う．

C 言語の for 文は，while 文を用いた記述に置き換えることができる．例えば，次の for 文を考える．

`for (` E_1 `;` E_2 `;` E_3 `)` S

ここで，E_1 は初期化式，E_2 は終了条件判定式，E_3 は更新式，S は for 文の本体の文である☆．上の for 文と等価な処理を行う while 文を用いた記述は次のように書ける．

E_1`;`
`while (` E_2 `) {`
　　S
　　E_3`;`
`}`

この規則を適用して，課題 1 にて記述した C 言語プログラムを書きなおすと**図 2** のようになる．図 2 の

† E-mail: nobuya@cs.okayama-u.ac.jp
†† E-mail: sasakura@momo.cs.okayama-u.ac.jp
††† E-mail: sato@cs.okayama-u.ac.jp
本課題に関する問い合わせは，下記のアドレス宛へ
E-mail: jikken-haru-kadai@arc.cs.okayama-u.ac.jp
追加情報等は，下記 URL を参照
http://www.arc.cs.okayama-u.ac.jp/~nobuya/jikken_kadai/

☆ 式は expression, 文は statement なのでそれぞれ E，S と表記．

```
1   int
2   vsum(int *addr, int n) {
3       int sum = 0;
4       int i;
5
6       ___(2)___
7       while (___(3)___) {
8           ______(1)______
9           ___(4)___
10      }
11      return sum;
12  }
```

図 2 for 文を while 文に変換したプログラム

```
1   int
2   vsum(int *addr, int n) {
3       int sum = 0;
4       int i;
5
6       ___(2)___
7   _L1:
8       if (____(5)____) goto _L2;
9       ______(1)______
10      ___(4)___
11      goto _L1;
12  _L2:
13      return sum;
14  }
```

図 3 while 文を if/goto 文を用いた処理に変換したプログラム

空欄部分を埋めて，プログラムを完成させよ．

なお，while 文は「条件判定式の値が真（正確には 0 以外）の間，本体の処理を繰り返す」という意味である．次に while 文で記述されている繰り返しを if 文と goto 文を用いた記述に変換する．次の while 文を考える．

`while (` E `)` S

ここで，E は終了条件判定式，S は while 文の本体の文である．上の while 文は次の文の並びに変換できる．

$\texttt{\_L}_1$`:`
　`if (!(` E `)) goto` $\texttt{\_L}_2$`;`
　S
　`goto` $\texttt{\_L}_1$`;`
$\texttt{\_L}_2$`:`
　...

ここで，ラベル $\texttt{\_L}_1$ と $\texttt{\_L}_2$ は，変換の際に導入されたラベルである．

図 2 のプログラムに上の規則を適用し変換すると**図 3** に示すプログラムとなる．図 3 に示すプログラムの空欄部分を埋めて，プログラムを完成させよ．

また，処理の流れをフローチャートにしたものを**図 4** に示す．図 4 の空欄部分を埋めてフローチャートを完成させよ．

以下で図 3 のプログラムをアセンブリ言語のプログラムへと変換する．

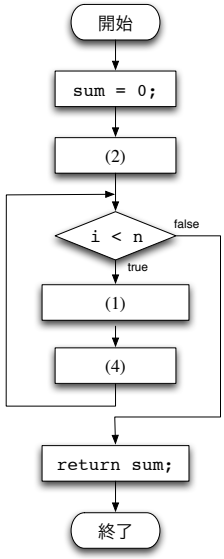

図 4 フローチャート

表 1 レジスタの割当

変数名	レジスタ	備考
addr	$a0	仮引数（配列の先頭アドレス）
n	$a1	仮引数 （データの要素数）
—	$v0	返値
sum	$s0	
i	$t0	
—	$t1	アドレス計算用
—	$t2	ロードしたデータを格納
—	$t3	繰り返しの終了判定用

2.2.2 レジスタの割当

C 言語プログラムコード中の各変数に対して，**表 1** に示すようにレジスタを割り当てる．

関数の本体部分冒頭で宣言されている 2 個の整数型変数 sum と i に，レジスタ $s0 と$t0 をそれぞれ割り当てる．加算対象のデータをメモリからロードする際のアドレスの計算用にレジスタ$t1 を，メモリからロードしたデータを一時的に格納するためにレジスタ $t2 を割り当てる．

2.2.3 アドレス計算

配列変数の要素の値の参照について考える．ここでは，バイト毎にアドレスが振られたバイトアドレッシングを想定する．配列変数 a の i 番目の要素の参照 $a[i]$ があった場合，a は配列データの格納されている先頭アドレスとなる．配列要素データ 1 個が占有するバイト数を s とすると，$a[i]$ のアドレスは，$a + s \times i$ で計算される（int 型の配列の場合 $s = 4$）．

アドレスを計算する際，まず，$s \times i$ の部分を計算し，結果を一時的にレジスタに格納する．ここで，2

```
1   vsum:
2           li      $s0, 0              # sum = 0;
3           li      $t0, 0              # i = 0;
4
5   _L1:
6           _________(6)_________       # $t3 = (i < n);
7           _________(7)___________     # if ($t3 == 0) goto _L2;
8
9           _________(8)_______         # $t1 = $t0 << 2;  ($t1 = $t0 * 4;)
10          _________(9)_________       # $t1 はデータのアドレス
11          ________(10)_______         # データをロードし，$t2 に格納
12          add     $s0, $s0, $t2       # 加算
13          addi    $t0, $t0, 1         # i++;
14          ___(11)____                 # goto _L1;
15  _L2:
16          move    $v0, $s0            # $v0 = $s0
17          ___(12)______               # return
```

図 5 関数 vsum のアセンブリ言語プログラム (vsum.s)

のべきの乗算においては，乗算命令ではなくシフト命令を利用する方法がよく用いられる．

次にベースアドレス（配列の先頭アドレス）である a を加算し，$a + s \times i$ で配列の該当要素データの格納されているアドレスを得る．このアドレスからデータをロードすることで，配列の該当要素を得る．

2.2.4 変換されたアセンブリ言語プログラム

変換されたアセンブリ言語プログラムを**図 5** に示す．図 5 の空欄部分を埋めてプログラムを完成させよ．

以下，図 5 について説明する．2 行目では総和結果を格納するレジスタ`$s0` の初期化を，3 行目ではループカウンタ用のレジスタ`$t0` の初期化をそれぞれ行う．

6〜7 行目は，繰り返しの条件判定処理である．繰り返し条件を満たさない場合，ループから脱出する．

9〜10 行目で，加算するデータの格納されているアドレスを計算する（計算方法は 2.2.3 にて説明）．

11 行目で，計算したアドレスからデータをロードしてレジスタ`$t2` に格納する．12 行目で，ロードした値を総和を格納するレジスタ`$s0` に加算する．13 行目では，ループカウンタ (レジスタ`$t0`) の値を 1 増やす（インクリメント）．14 行目で繰り返しの終了判定処理の部分（ラベル`_L1`）へジャンプする．

15 行目以降のコードは，終了処理である．16 行目で計算結果を返値用のレジスタ`$v0` に転送し，最後に 17 行目で戻りアドレスにジャンプし，この関数からリターンする．

3. (課題 3) コードサイズと実行ステップ数

3.1 問　　題

(課題 2) で作成したアセンブリ言語プログラムのコードサイズを求めよ．また，プログラムを実行する際の実行ステップ数を求めよ．

今回，MIPS アセンブリ言語の擬似命令を用いているが，擬似命令も 1 命令とカウントするものとする．また，実行ステップ数の算出においては，配列データの要素数は n として，n を用いた式で示すこと．

3.2 ヒ　ン　ト

コードサイズに関しては，今回の対象マシンである MIPS は RISC プロセッサであるため，1 命令は 1 ワードすなわち 4 バイトとしてカウントする．

実行ステップ数の算出に関しては，関数が 1 回呼ばれた際に

(1) 固定的に実行される命令数 c と
(2) データ要素数 n に依存して繰り返し実行される命令数 v

をそれぞれ数え上げることで，$c + n \times v$ として求めることができる．

4. (課題 4) 最適化

4.1 問　　題

(課題 2) で変換したアセンブリ言語プログラムは，元の C 言語プログラムをほぼ機械的に変換しただけで，コードサイズ，プログラム実行速度の点で優れているとは言い難い．どの部分に最適化の余地があるかを検討し，プログラムの実行速度を上げるための最適化，あるいはコードサイズを小さくする最適化を施したアセンブリ言語プログラムを示せ．

また，コードサイズとデータ要素数が n の場合の実行ステップ数を示し，どの程度のステップ数削減あるいはコードサイズの削減が達成出来ているかを示せ．

参　考　文　献

1) David. A. Patterson, John L Hennessy: コンピュータの構成と設計— ハードウエアとソフトウエアのインタフェース — 第 4 版（上巻），(成田光彰訳) 日経 BP 社, 2011.