ルビス　ルスファン　アンシャー

15\_15953

**課題１（構造体の配列の並べ替え）**

ソースコード名：sort.c

目的：関数ポインターを用いて、配列を並べ替える。

プログラムは5個の関数から成り立っている。

最初にhdataという構造体を宣言する。

typedef struct{

int id;

float height;

} hdata;

swap関数は２つのhdataの順番を交換する

void swap(hdata \*x, hdata \*y){

hdata temp;

temp = \*x;

\*x = \*y;

\*y = temp;

}

compar\_idは2個のhdataのidを比較する関数

int compar\_id(const hdata \*a, const hdata \*b){

if ((a -> id) < (b -> id)){

return 1;

} else return -1;

}

compar\_heightは2個のhdataのheightを比較する

int compar\_height(const hdata \*a, const hdata \*b){

if ((a -> height) < (b -> height)){

return 1;

} else return -1;

}

sort関数はhdataの配列、配列の長さと関数ポインタを受け取り、関数ポインターによって、配列をソートする。compar\_idを受け取れば、idを昇順にソートし、compar\_heightを受け取る場合、heightを昇順にソートする。

int sort(hdata \*base, int num, int (\*compar)(const hdata \*a, const hdata \*b)){

int i, j;

//ソートはバブルソートである

for (i = 0; i < num; i++){

int k = num - 1;

for (j = 0; j < k; j++){

//base[j+1]はbase[j]より小さいとき、スワップする

if (compar(&base[j+1], &base[j]) == 1){

swap(&base[j+1], &base[j]);

}

}

k = k - 1;

}

return 0;

}

main関数で配列を定義して、ソートを実行する。その後、実行結果を表示する。

int main(){

hdata base[] = {

{4, 154},

{2, 166},

{8, 124},

{1, 176}

};

sort(base, 4, compar\_id);

printf("compare id\n");

for (int i = 0; i < 4; i++){

printf("%d:%d(id=%f)\n",i+1, base[i].id, base[i].height);

}

printf("\n");

sort(base, 4, compar\_height);

printf("compare height\n");

for (int i = 0; i < 4; i++){

printf("%d:%d(id=%f)\n",i+1, base[i].id, base[i].height);

}

return 0;

}

**課題２（bltの実装）**

ソースコード名：blt.s

.data

A:

.asciiz "true"

B:

.asciiz "false"

.text

main:

li $t1, 4

li $t2, 6

slt $t0, $t1, $t2 # $t1 < $t2をチェックする

bne $t0, $zero, label # $t0 != 0であれば、labelにジャンプする

li $v0, 4

la $a0, B

syscall # falseをprint

j end

label:

li $v0, 4

la $a0, A

syscall # trueをprint

end:

jr $ra

**課題3（２つの配列の要素の和）**

ソースコード名：addition.s

.data

A:

.word 1 2 3 4

B:

.word 5 6 7 8

.text

main:

li $t0, 0 #Aのインデックスは先頭(0)から始まる

li $t1, 12 #Bのインデックスは末尾(12)から始まる

# wordの長さは4なので、足し算の回数を数えるために4を$t6に入れる。

li $t6, 4

loop:

beq $t6, $zero, end # $t6が0になれば、足し算が終わる

lw $t3, A($t0)

lw $t4, B($t1)

add $t5, $t3, $t4

#足し算の結果を表示する

li $v0, 1

move $a0, $t5

syscall

#Aのインデックスを1個前に進み、Bのインデックスが1個後ろ動く

addi $t0, $t0, 4

addi $t1, $t1, -4

addi $t6, $t6, -1

j loop

end:

jr $ra

**課題4（最大値の検索）**

ソースコード名：max.s

.data

A:

.word 2 7 113 1 3 9

.text

main:

li $t0, 0 # 最大初期値を0にする

li $t1, 0 # インデックスのためのレジスター

loop:　(最大値を求めるために、wordをループでチェックする)

lw $t2, A($t1) # 配列からの値を$t2に入れる

beq $t2, $zero, end # 値が0ならば、配列のチェックが完了

# $t0と$t2を比較し、$t2が大きい場合、スワップする

blt $t0, $t2, change

addi $t1, $t1, 4

j loop

change: # $t0に新しい最大値を代入

move $t0, $t2

addi $t1, $t1, 4 # インデックスを４で足す

j loop

end:

# 最大値を表示する

li $v0, 1

move $a0, $t0

syscall

j $ra

**課題5（bleの実装）**

ソースコード名：ble.s

.data

A:

.asciiz "true"

B:

.asciiz "false"

.text

main:

li $t0, 4

li $t1, 3

sub $t2, $t0, $t1 # $t2 = $t0 - $t1

# $t2 < 1をチェックした後、$t3にその結果($t2 < 1の場合、$t3 = 1, $t2 !< 1の場合、$t3 = 0)を代入する。$t3は0の場合、falseを表示して、そうでない場合、trueを表示。

slti $t3, $t2, 1

bne $t3, $zero, label

li $v0, 4

la $a0, B

syscall # print false

j end

label:

li $v0, 4

la $a0, A

syscall # print true

end:

jr $ra