**Introduction**

A Sobel filter is an application used in image processing and computer vision, particularly within edge detection algorithms where it creates an image emphasising edges. It uses two 3X3 kernels which are convoluted with the original image to calculate approximations of the derivatives – one for horizontal changes, and one for vertical. The C code below only calculates the horizontal gradient. This code results in many processor instructions.

#define WIDTH 32

#define HEIGHT 32

void main(void)

{

int a, b, result;

unsigned char \*buf\_i = (unsigned char\*)0x401000, \*buf\_o = (unsigned char\*)0x402000;

for (a = 1; a < HEIGHT - 1; a++)

{

for (b = 1; b < WIDTH - 1; b++)

{

/\* Horizontal Gradient \*/

result=(

1\*(int)buf\_i[(a - 1) \* WIDTH + b - 1] +

2\*(int)buf\_i[(a - 1) \* WIDTH + b ] +

1\*(int)buf\_i[(a - 1) \* WIDTH + b + 1] +

0\*(int)buf\_i[ a \* WIDTH + b - 1] +

0\*(int)buf\_i[ a \* WIDTH + b ] +

0\*(int)buf\_i[ a \* WIDTH + b + 1] +

-1\*(int)buf\_i[(a + 1) \* WIDTH + b - 1] +

-2\*(int)buf\_i[(a + 1) \* WIDTH + b ] +

-1\*(int)buf\_i[(a + 1) \* WIDTH + b + 1] );

// Clipping Operation

if(result < 0) buf\_o[a \* WIDTH + b] = 0;

else if (result > 255) buf\_o[a \* WIDTH + b] = 255;

else buf\_o[a \* WIDTH + b] = result;

}

}

}

To reduce the number of instructions, the C code is modified to use two custom instructions (hgradient1, hgradient2), giving the following C code:

#define WIDTH 32

#define HEIGHT 32

#define hgradient1(p, q) ((p) + ((q) - \*(int \*) 0x12344321)) //HEX Code: 0x32

#define hgradient2(p, q) ((p) + ((q) + \*(int \*) 0x12344321)) //HEX Code: 0x30

void main(void)

{

int a, b, result;

int temp1,temp2;

int p, q, r;

int max = 255;

unsigned char \*buf\_i = (unsigned char\*)0x401000, \*buf\_o = (unsigned char\*)0x402000;

for (a = 1; a < HEIGHT - 1; a++)

{

for (b = 1; b < WIDTH - 1; b++)

{

/\* Horizontal Gradient \*/

p=(int)buf\_i[(a - 1) \* WIDTH + b - 1];

q=(int)buf\_i[(a - 1) \* WIDTH + b];

r=(int)buf\_i[(a - 1) \* WIDTH + b + 1];

temp1= hgradient1(q,r); // 2\*q + r

temp1= p + temp1;

p=(int)buf\_i[(a + 1) \* WIDTH + b - 1];

q=(int)buf\_i[(a + 1) \* WIDTH + b];

r=(int)buf\_i[(a + 1) \* WIDTH + b + 1];

temp2= hgradient1(q,r); // 2\*q + r

temp2= p + temp2;

buf\_o[a \* WIDTH + b] = hgradient2(temp1, temp2); // clip(temp1 - temp2)

}

}

}

#### Assignment

Add hardware support for the custom instructions (**hgradient1** and **hgradient2**) in the current mMIPS implementation by modifying all or some of the provided files (ctrl.v, aluctrl.v and alu.v).

You can infer the functionality of hgradient1 and hgradient2 by comparing the C code above. The important information is summarised below:

##### hgradient1

Instruction function code 0x32

Equivalent C function:

int hgradient1(int q, int r) {

return (2 \* q) + r;

}

##### hgradient2

Instruction function code 0x30

Equivalent C function:

int hgradient2(int temp1, int temp2) {

int result = temp1 - temp2;

if (result > 255) {

result = 255;

} else if (result < 0) {

result = 0;

}

return result;

}