LAB3: Embedded Software

1&2.in this course we use the semaphore and the MSG Queue to schedule the task grayscale and assic.(As for the task2 we should changed to force the format into integer format)

The actor graySDF could be transfering the original picture to the grayed picture

In 3 groups like seminar in float format.

```
void graySDF(unsigned char* orig,unsigned char* after)
{
    int sizeX=orig[0];
    int sizeY=orig[1];
    int fullsize=sizeX*sizeY;
    int i,j;
    //unsigned char grayscale_img[fullsize+3];
    unsigned char* imgP;
    unsigned char* share;
    after[0]=sizeX;
    after[1]=sizeY;
    after[2]=orig[2];
    for(i=0;i<fullsize;i++)
    {
        after[i + 3]= (unsigned char)( 0.3125 * orig[3 * i + 3 ] + 0.5625 * orig[3 * i + 4] + 0.125
    }
    imgP = after;
    share = (unsigned char*) SHARED_ONCHIP_BASE;
    for (i=0;i<fullsize+3;i++)
    {
        *share++ = *imgP++;
    }
}
```

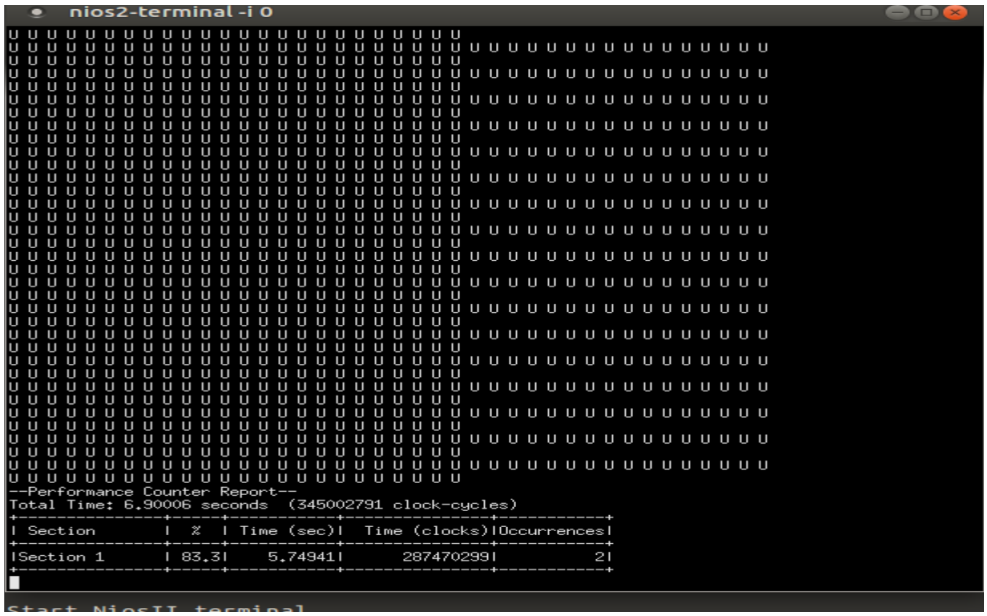
As for the integer part we force the processed picture into the integer format by multiple integer and divide the integer

```
void asciiSDF(unsigned char* orig)
{
int sizeX = orig[0], sizeY = orig[1];
int i,j;
int fullsize=sizeX*sizeY;
unsigned char ASCII[sizeX * sizeY+3];
char asciiLevels[16] = {' ','.',':','-','=','+','/','t','z','U','w','*','0','#','%','@'};
unsigned char* imgP;
unsigned char* share;
ASCII[0] = sizeX ;
ASCII[1] = sizeY ;
ASCII[2] = orig[2] ;
for( i = 0; i < sizeX * sizeY; i++)
{
ASCII[i+3] = asciiLevels[((orig[i+3])/16)];
}
for( i = 0; i < sizeY; i++)
    {
    for( j=0; j< sizeX;j++)
        {
        printf("%c ",ASCII[j+(i*sizeX)+3]);
        }
    printf("\n");
    }
        imgP = ASCII;
    share = (unsigned char*) SHARED_ONCHIP_BASE;
    for (i=0;i<fullsize+3;i++)
    {
        *share++ = *imgP++;
    }
}
```

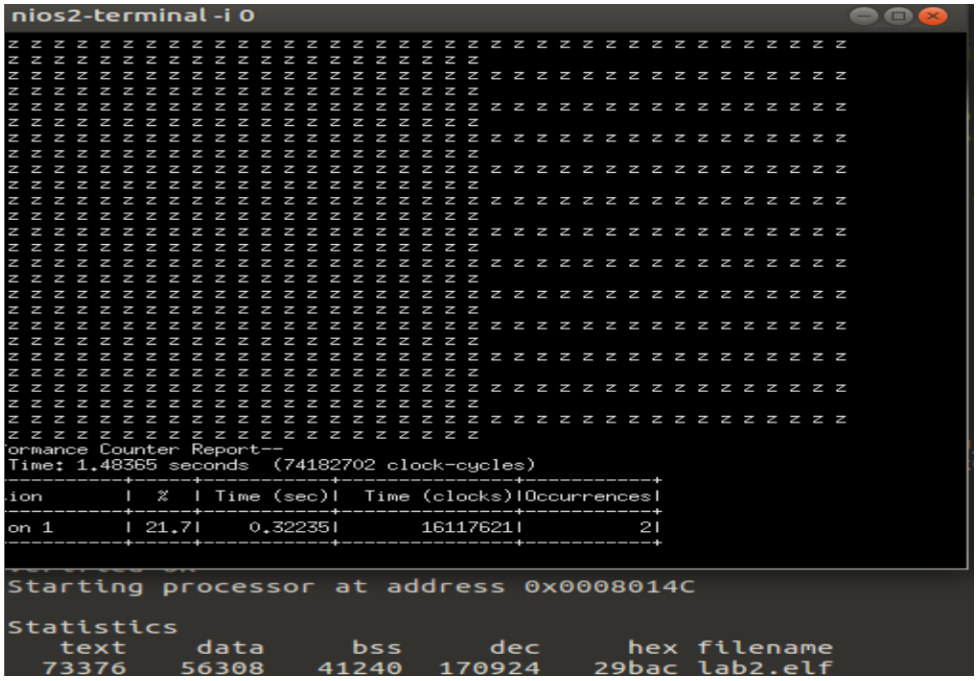
In Ascii part, we should transfer the grayed picture to assic. We write the assic function like the following to transfer the grayed picture into assic ones.

AS for the message queue part, the idea is that first we should use **OSQCreate** to create the queue task. **OSQPend** to sends a message to a task through a queue. And the **OSQPost** to receive the task from the queue.

The result of the float and the integer is like following.

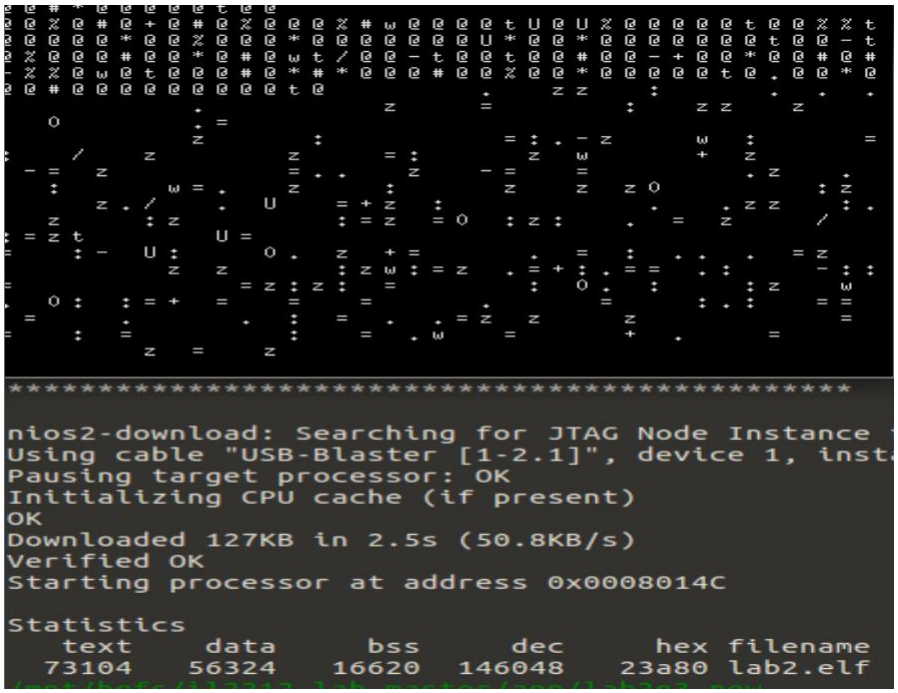


(the result of the float format graySDF)



(the result of the integer format graySDF)

3.4.to just write the bare we just need to write the actor in schedule so do the resize part. the resize part is like following:

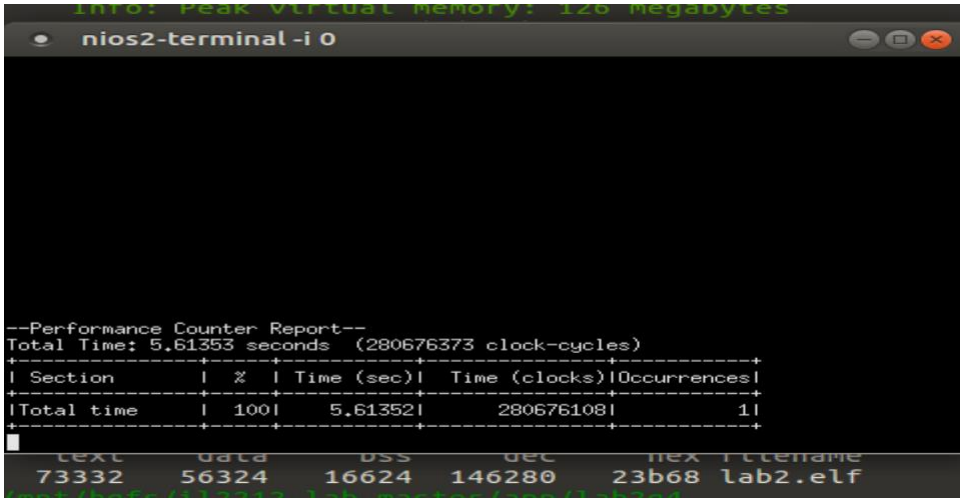


The result is like above.

```
void resize(unsigned char* img) {
   int maxX=img[0];// width of the picture
   int maxY=img[1];//Height of the picture
    unsigned char* output_image = (unsigned char*) malloc((maxX/2) * (maxY/2) * sizeof(unsigned char)
    int i, j, k, l;
    for (i = 0; i < maxY; i += 2) {
    for (j = 0; j < maxX; j += 2) {
    int sum = 0;
        for (k = i; k < i + 2; k++) {
        for (l = j; l < j + 2; l++) {
        sum += img[k * maxX + l];}
}
        output_image[(i / 2) * (maxX / 2) + (j / 2)] = sum / 4;


}
}
}
```

And the result is like the following:



5. we use the mutex to control the whole task in schedule. But it has the problem of the

overflow. Due to the limitation of the memory, we could not choose the 64*64 picture. In the contract, we will choose the 32*32 image. However, to let every mode run correctly, we could use the flag to control it.

And the result is like that:

And we assign the task like the following:cpu_0 to graySDF cpu_1 to resize cpu_2 to assic