##
Project: Bump_it_up
Autors:
Marco Espinoza
Jose Campos
##

This project is uploaded in the following git repostery:
https://github.com/mespinoza86/mespinoza_jcampos_embedded_2017/tree/develop

It consists on develop a Yocto Meta-Layer with some recipes capable to convert a rgb image into a yuv image.
Inside the meta-tec layer there are 3 important recipes:
- recipe-rgb2yuv_c
- recipe-intrinsicrgb2yuv
- recipe-neonrgb2yuv

The command line to compile this layer is the following:

bitbake meta-tec-image

The explanation for each of the recipe developed is:

*recipe-rgb2yuv:

It has a program developed in C, and the main function is convert a rgb image to yuv.
All the methods used are from C and it is not using neon to do the operations.

It was measured the time expended by this program to do the image conversion and it was: 338000 us for a image.rgb.

* recipe-intrinsicrgb2yuv:

It has a program developed in C but using the neon intrinsic library to do the image conversion from rgb to yuv.
All the methods used were written in C language but using the intrinsic vectors and operations from the neon lib.

It was measured the time expended by this program to do the image conversion and it was: 156902 us for a image.rgb.

* recipe-neonrgb2yuv:

It has a program developed in C but using the neon library to do the image conversion from rgb to yuv.
The main method to do the conversion was written using ARM neon asm
It allowed to have an improvement in the image conversion, due to it was used the registers directly in asm code.

It was measured the time expended by this program to do the image conversion and it was: 138085 us for a image.rgb.

The algorithm used to convert the image from rgb to yuv is the following:

Formula to calculate the YUV from RGB:
Y = ((66xR + 129xG + 25xB + 128) >> 8) + 16
U = ((-38xR - 74xG + 112xB + 128) >> 8) + 128
V = ((112xR - 94xG - 18xB + 128) >> 8) + 128

For each RGB pixels, the Y, U and V are calculated with the equation showed.
The whole pixels belonging to the image are calculated, and they are stored in 3 different buffers (Y, U and V).
To generate the new image, the format YUV422 is used, it is the following:

yuv[0] = u[0]
yuv[1] = y[0]
yuv[2] = v[1]
yuv[3] = y[1]
.
.
.

It means, all the Y are stored in the new image, but only the pair U and the impair V are stored, it generate the format called YUV422, because from 6 RGB bytes, only 4 YUV bytes are stored.

This algorithm was implemented using C, intrinc and neon.

Conclusions:

* It was possible to developed 3 programs to do the image conversion from rgb to yuv, allowing to identify the differences between the C programming and the improvement generated by neon in the program execution.

* In embedded systems it is very important to develop programs with good performance due to the memory and battery are limited in this kind of systems, for this reason the implementation of libraries which help to use appropriately the hardware architecture is very important.

* In this project it was demonstrated despite C programming is a low programming level, the performance still could be improved if the program is written with asm code and proper vector instruction.

* The timing measurement demonstrated the neon_rgb2yuv program obtained the better performance, however the intrinsic had a similar result, without program in asm code, and the rgb2yuv program written in C language obtained the worst performance.