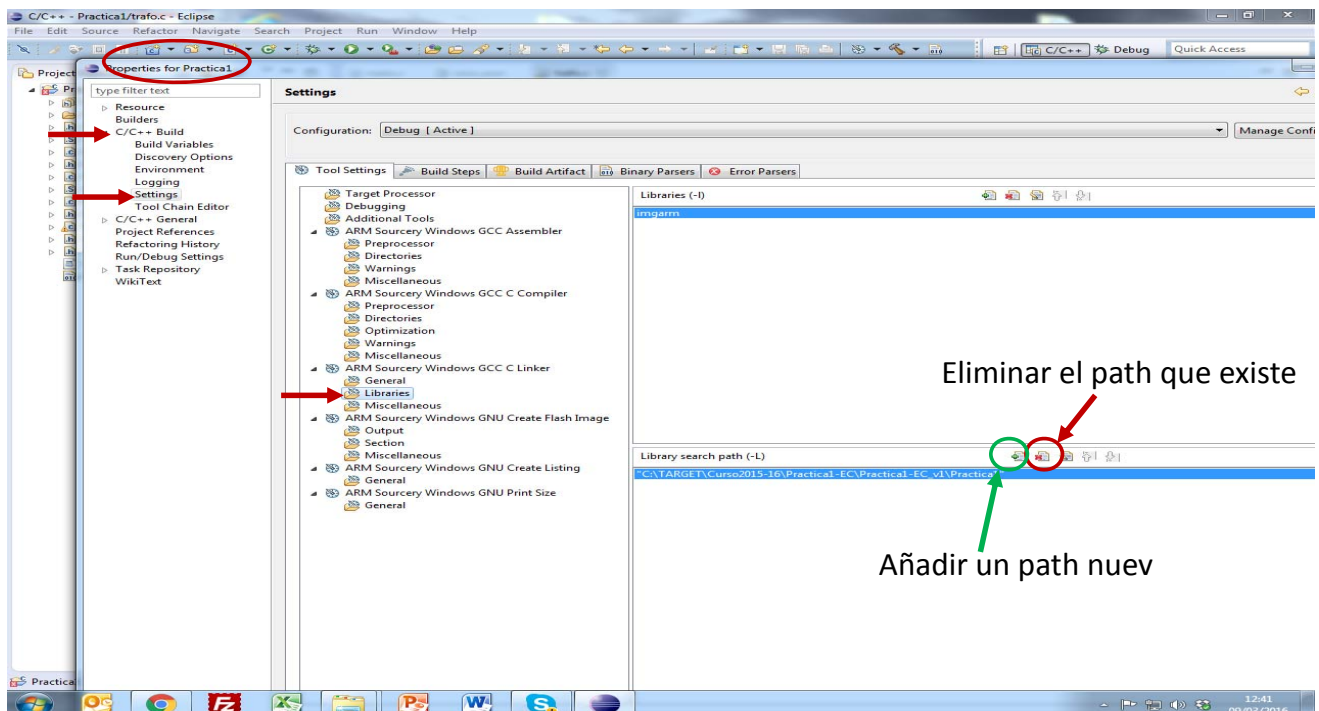


# Cosas importantes sobre la práctica 1

**Tenéis que indicar el path de la biblioteca libimgarm.a**

- Ratón encima del proyecto: Botón derecho seleccionar propiedades



## Sobre las funciones que hay que hacer en ensamblador

- En la función `apply-gaussian()` hay que tener en cuenta que se llama a otra función que tiene 5 parámetros:

`int gaussian(unsigned char* im1, int width, int height, int i, int j)`

**El quinto parámetro hay que pasarlo por la pila**

- En la función `rgb2gray()` hay que multiplicar por coeficientes

`dest = (3483*orig.R + 11718*orig.G + 1183*orig.B) / 16384`

Estos coeficientes tienen valores muy grandes que no permite la `inst mov`, luego para guardar en un registro ese valor hay que usar `ldr`:

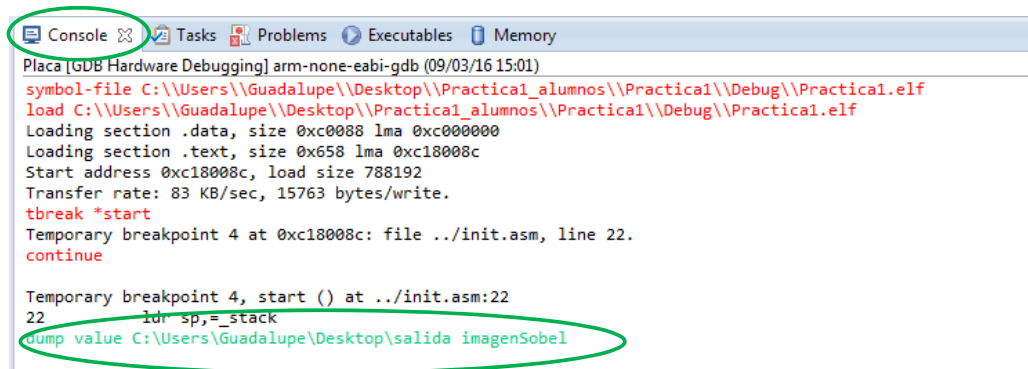
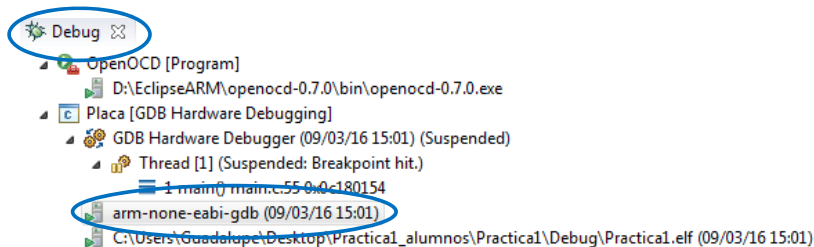
~~`mov, R0, # 3483`~~

`ldr R0, = 3483`

## Para Visualizar las imágenes que se van generando

### 1. Pasar los valores de la memoria de la placa a un fichero en el PC

- Poner un break point en la instrucción `return` del `main`
- En la ventana `Debug` seleccionar: **arm-none-eabi-gdb**
- En la ventana de la consola escribir: **dump value nombreFicheroSalida NombreMatrizImagen**

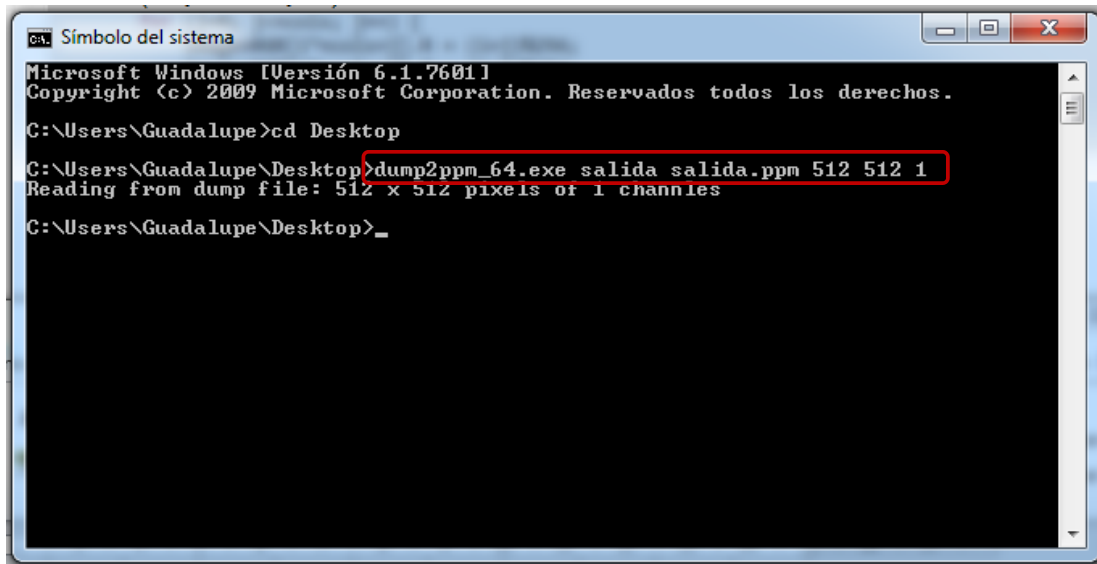


## Para Visualizar las imágenes que se van generando

### 2. Convertir el **fichero de salida** de la práctica a **extensión ppm**

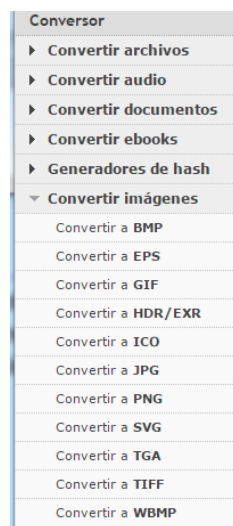
- Abrir la consola del sistema
- Colocarse en la carpeta donde está el fichero de salida de la práctica y el dump que se proporciona con la práctica (de 32 o 64 bits dependiendo del ordenador)
- Desde esa carpeta escribir en la consola

**dump2ppm.\_64.exe nombreFicheroSalida nombrefichero.ppm 512 512 1**



### Para **abrir un fichero** con extensión **ppm**

- Si en el ordenador no se dispone de ninguna herramienta para visualizar en ese formato, **existen herramientas online para convertir una imagen PPM en BMP**, formato propietario de Windows.
- Por ejemplo



### Convierte una imagen al formato BMP

Free Image Download  
1,000,000 Free Photos Online Sign Up For Free Hi-Res

Convertor de imágenes en línea

Convierte tus imágenes al formato BMP con este convertor de imágenes online gratuito. Sube tu archivo y selecciona como opción los efectos que quieras aplicar para cambiar tu imagen.

**Carga la imagen que deseas convertir a BMP:**

**Seleccionar archivo** Ningún archivo seleccionado

**O introduce la URL de la imagen que deseas convertir a BMP:**

(e.g. <http://bit.ly/b2dlVA>)

**O selecciona un archivo de tu nube de almacenamiento para una conversión a BMP:**

Choose from Dropbox Choose from Google Drive

Ajustes opcionales

Modificar tamaño: [ ] pixeles x [ ] pixeles

Color: ☒ Coloración ☐ Gris ☐ Monocromo

Negativo

☐ Año 1980 ☐ Año 1900

Mejorar: ☐ Ecualizar ☐ Normalizar ☐ Mejorar

☐ Enfocar ☐ Antialiasing ☐ Eliminar manchas

DPI: [ ]

**Convertir archivo**

(al hacer clic confirmas que comprendes y aceptas nuestras

## Para Visualizar las imágenes que se van generando

- Esta es la imagen final que genera la práctica



## Para saber las direcciones de memoria de las variables y de las funciones

- En esta práctica al tener variables que ocupan mucho (Lena es una imagen de 512x512) no es fácil saberlo mirando la ventana de desensamblado
- **Para saber la dirección donde empiezan las funciones**
  - Poniendo el ratón encima de la función te dice en qué dirección de memoria está

```
main.c misc.asm trafo.c init.asm
imagenRGB = (pixelRGB*) imageLena512;
}
#endif

int main() {
    short int time = 0;
    timer_init();
    initRGB(N,M);
    timer_start();
    RGB2GrayMatrix(imagenRGB, imagenGris, N, M);
    RGB2GrayMatrix = (void (pixelRGB *, unsigned char *, int, int)) 0xc180318 <RGB2GrayMatrix>
    apply_sobel(imagenoduss, imagen300x1, N, M);
    time = timer_stop();
    return 0;
}
```

# Para saber las direcciones de memoria de las variables y de las funciones

- **Para saber la dirección de las variables:**

- Poner un break point en las funciones
- Cuando se para en una función dar al icono de entrar en la función una vez
- Una vez dentro de la función hay dos maneras de saber cuales son las direcciones de los vectores que contienen las imágenes
  - En los registros se puede ver los valores de los parámetros de la función
    - En las tres funciones que tenemos los primeros parámetros son las direcciones de los vectores que contienen las imágenes
  - Poniendo el ratón encima de las variables de la función se puede ver su dirección y su contenido

The screenshot shows a debugger interface with the 'Registers' window open. The registers are listed as follows:

Name	Value
r0	0xc0c008c
r1	0xc10008c
r2	512
r3	512
r4	0

The code editor shows the function `void apply_sobel(unsigned char im1[], unsigned char im2[], int width, int height)`. The parameter `im1` is circled in red. A red arrow points from the 'Registers' window to the `im1` parameter in the code.