# Day 6-1

*Assignment:*

- Finish up labs 4-7, Due Thur, 1-Oct
- Project Proposal, Due Tuesday
- HW 05, Due Tuesday

*Today's Topics:*

- Raspberry Pi, IoT
- Project Proposal
- GPIO via mmap()
- Boot up

# Raspberry Pi & IoT

- Who's signing up?

# Lab Writeup

**What to turn in**

Make a subdirectory in your github repository called **lab07**. Do this for each lab partner.

Put all your files in the directory, include a **ReadMe.txt** which will serve as your lab report. You may share code with your lab partner, but you must write your own ReadMe.txt. This report should use a memo format (Google "memo format" for examples) that contains:

- To, From, Date and Subject fields.
- You and your lab partner's names.
- A sentence or two giving an introduction to what the lab was about.
- A brief section of each part of the lab noting what you did and referring to any code by filename.
- A sentence or two in conclusion giving your thoughts on the lab.

Document your code.

# Project Topics

1. Automated Lock System using
    1. Voice Recognition
    2. Secret Knock Detection
2. Intruder alerting system
3. Automated table tennis ball dispenser
4. Facebook Like Counter
5. Video Capture and Image Processing, Image Recognition
6. Remote Desktop Control
7. Installing another Linux Distro
8. Using Microsoft Xbox Kinect with BeagleBone Black

1. Home energy management system using multiple passive IR sensors.
2. Some equipment in labs, which usually on high demand can be monitored by bone and can notify people when it is free, also different user's usage can be collected and a log can be maintained.
3. Self-balancing skateboard
4. Accident detection and messaging system using gps and gsm

# Project Topics

1. CONTROLLING HOME APPLIANCE
2. MOTION OF QUAD COPTER ON ANY TRAJECTORY
3. EYE WRTITER
4. PLOTTING TIME

1. Weather
    1. Calculation of Wind Speed Velocity
    2. Temperature Sensor
    3. Bone can monitor weather using sensors and *cross reference* it with information from internet. Then it can notify you about it.

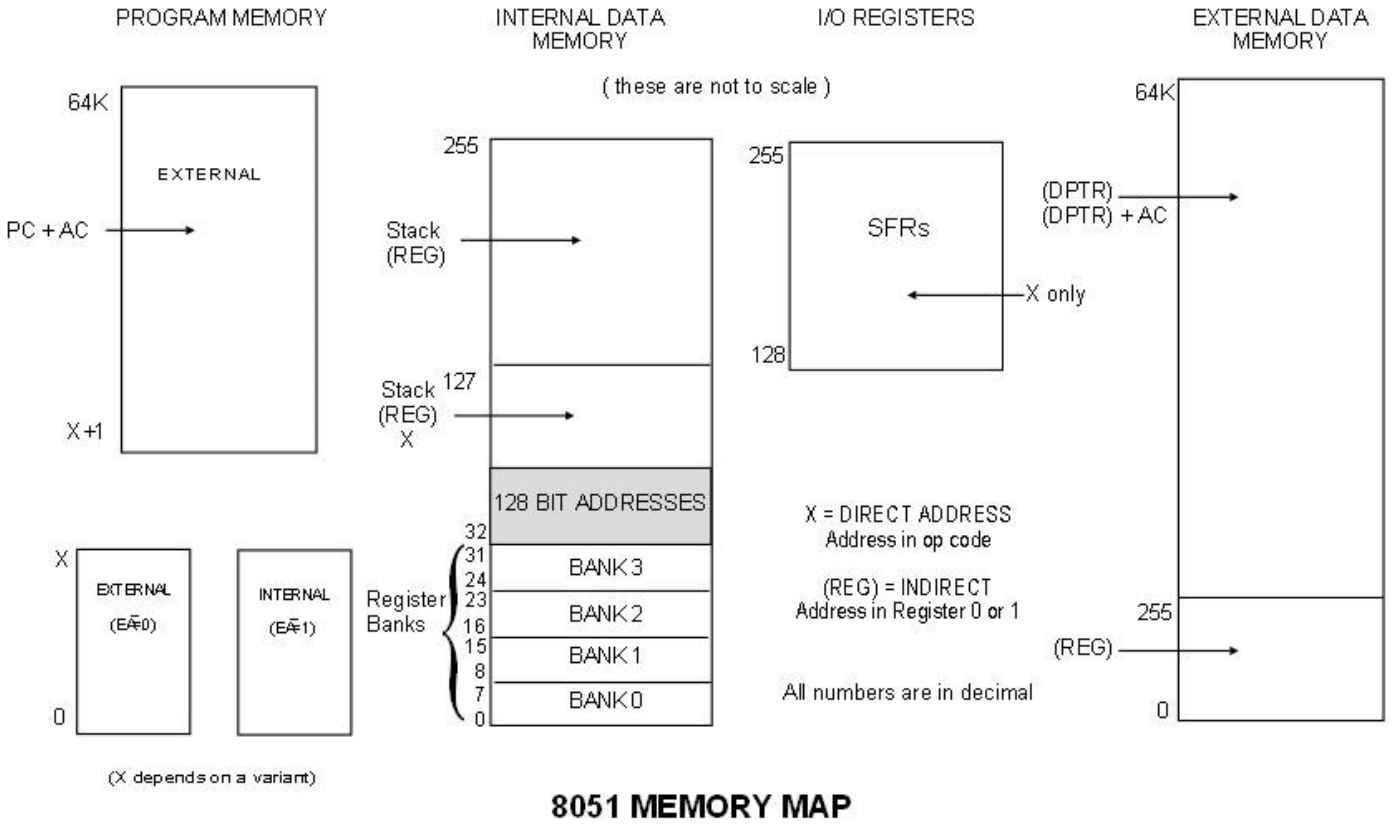
# Project Proposal

- Example in Google Labs folder (https://drive.google.com/drive/u/0/folders/0B5UNMAgIJB74fkZtSFIzUHd5RG9vRm5HZUVuV3E4bGpMQUNMWXBfbENHZUVxRTd2dU5HU0k)
- Write your own proposal
  - *Executive Summary*: A short paragraph stating what the project is about.
  - *Current State*: Note if this project is building on a previous project. If so, try running the software on your Bone and note how it works.
  - *Team Structure*: Your project should be small enough to require no more than four team members. Note what roles they will play in the project. Note their names if you know who you want.
  - *Equipment Needed*: Do you need equipment beyond what you already have? If so, list it with an estimated cost and a link to where it can be ordered.
- Due Tues 29-Sept-2015

# 07-1 GPIO via mmap()

# Memory Maps

- A table/diagram that shows how memory is laid out




**8051 MEMORY MAP**

http://www.computer-solutions.co.uk/info/micro-search/8051/8051_tutorial.htm

# Memory Maps

- A table/diagram that shows how memory is laid out
- MSP430

| Address | Type of memory |
|---|---|
| 0xFFFF | interrupt and reset |
| 0xFFC0 | vector table |
| 0xFFBF | flash code memory |
| 0xF800 | (lower boundary varies) |
| 0xF7FF | |
| 0x1100 | |
| 0x10FF | flash |
| 0x1000 | information memory |
| 0x0FFF | *bootstrap loader* |
| 0x0C00 | (*not in F20xx*) |
| 0x0BFF | |
| 0x0280 | |
| 0x027F | RAM |
| 0x0200 | (upper boundary varies) |
| 0x01FF | peripheral registers |
| 0x0100 | with word access |
| 0x00FF | peripheral registers |
| 0x0100 | with byte access |
| 0x000F | special function registers |
| 0x0000 | (byte access) |

https://soumyageorgek.files.wordpress.com/2010/11/screenshot-3.png

# GPIO via sysfs

- So far we've been access the GPIO pins via sysfs
- You can turn a USR LED on with

```
bone$ cd /sys/class/leds/beaglebone\:green\:usr3
bone$ echo none > trigger
bone$ echo 1 > brightness
```

- sysf is portable, but can be slow
- What if speed is needed?

# GPIO via mmap()

- All the IO on the am335x is memory mapped
- You can look them up on the am335x Technical Reference Manual (TRM)

# USR3 LED

```
bone$ cd ~/exercises/gpio
bone$ ./findGPIO.js USR3
{ name: 'USR3', gpio: 56, led: 'usr3',
mux: 'gpmc_a8', key: 'USR3',
muxRegOffset: '0x060',
options: [ 'gpmc_a8', 'gmii2_rxd3',
'rgmii2_rd3', 'mmc2_dat6', 'gpmc_a24',
'pr1_mii1_rxd0', 'mcasp0_aclkx', 'gpio1_24' ] }
USR3 (gpio 56) mode: 7 (gpio1_24) 0x060 pullup
pin 24 (44e10860): (MUX UNCLAIMED) (GPIO
UNCLAIMED
```

# From Table 2-3 of TRM

| | | | | |
|---|---|---|---|---|
| DMTIMER7 | 0x4804_A000 | 0x4804_AFFF | 4KB | DMTimer7 Registers |
| | 0x4804_B000 | 0x4804_BFFF | 4KB | Reserved |
| GPIO1 | 0x4804_C000 | 0x4804_CFFF | 4KB | GPIO1 Registers |
| | 0x4804_D000 | 0x4804_DFFF | 4KB | Reserved |
| Reserved | 0x4804_E000 | 0x4804_FFFF | 8KB | Reserved |




- Base address is **`0x4804_C000`.**
- Click on GPIO1

Table 25-5. GPIO REGISTERS

| Offset | Acronym | Register Name | Section |
|---|---|---|---|
| 0h | GPIO_REVISION | | Section 25.4.1.1 |
| 10h | GPIO_SYSCONFIG | | Section 25.4.1.2 |
| 20h | GPIO_EOI | | Section 25.4.1.3 |
| 24h | GPIO_IRQSTATUS_RAW_0 | | Section 25.4.1.4 |
| 28h | GPIO_IRQSTATUS_RAW_1 | | Section 25.4.1.5 |
| 2Ch | GPIO_IRQSTATUS_0 | | Section 25.4.1.6 |
| 30h | GPIO_IRQSTATUS_1 | | Section 25.4.1.7 |
| 34h | GPIO_IRQSTATUS_SET_0 | | Section 25.4.1.8 |
| 38h | GPIO_IRQSTATUS_SET_1 | | Section 25.4.1.9 |
| 3Ch | GPIO_IRQSTATUS_CLR_0 | | Section 25.4.1.10 |
| 40h | GPIO_IRQSTATUS_CLR_1 | | Section 25.4.1.11 |
| 44h | GPIO_IRQWAKEN_0 | | Section 25.4.1.12 |
| 48h | GPIO_IRQWAKEN_1 | | Section 25.4.1.13 |
| 114h | GPIO_SYSSTATUS | | Section 25.4.1.14 |
| 130h | GPIO_CTRL | | Section 25.4.1.15 |
| [illegible] | GPIO_OE | **0x4804_c000 + 13c = 0x4804_c13c** | Section 25.4.1.16 |
| 138h | GPIO_DATAIN | Address for GPIO_DATAOUT | Section 25.4.1.17 |
| 13Ch | GPIO_DATAOUT | | Section 25.4.1.18 |
| [illegible] | GPIO_LEVELDETECT0 | | Section 25.4.1.19 |
| 144h | GPIO_LEVELDETECT1 | | Section 25.4.1.20 |
| 148h | GPIO_RISINGDETECT | | Section 25.4.1.21 |
| 14Ch | GPIO_FALLINGDETECT | | Section 25.4.1.22 |
| 150h | GPIO_DEBOUNCENABLE | | Section 25.4.1.23 |
| 154h | GPIO_DEBOUNCINGTIME | | Section 25.4.1.24 |
| 190h | GPIO_CLEARDATAOUT | | Section 25.4.1.25 |
| 194h | GPIO_SETDATAOUT | | Section 25.4.1.26 |

# devmem2

- A program that reads/writes any memory location

```
bone$ devmem2 0x4804c13c
/dev/mem opened.
Memory mapped at address 0xb6f99000.
Read at address 0x4804C13C (0xb6f9913c): 0x01800000
```


***gpio1_24***

Bit 24 shows that status of the LED

```
bone$ wget http://free-electrons.com/pub/mirror/devmem2.c
bone$ gcc -o devmem2 devmem2.c
```

# Set the LED - microcontroller

- The PIC way
  - Read register
  - OR with (1<<24)
  - Write register
- 3 operations

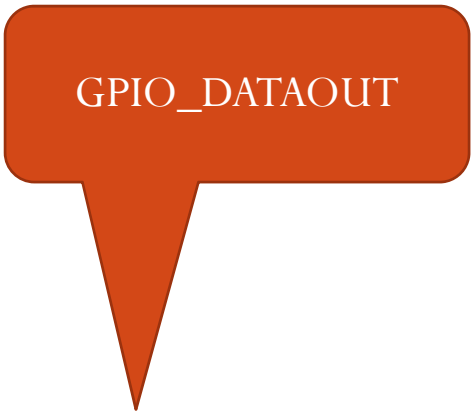
# Toggle the LED

- Use **GPIO_SETDATAOUT** and **GPIO_CLEARDATAOUT**

| | | |
|---|---|---|
| 154h | GPIO_DEBOUNCINGTIME | Section 25.4.1.24 |
| 190h | GPIO_CLEARDATAOUT | Section 25.4.1.25 |
| 194h | GPIO_SETDATAOUT | Section 25.4.1.26 |

- Write to **GPIO_SETDATAOUT** a value with 1's for the pins to be set to 1
- Write to **GPIO_CLEARDATAOUT** a value with 1's for the pins to be cleared to 0
- Use 0x190 to Clear

# Turn LED off then on




- Off

```
bone$ devmem2 0x4804c190 w 0x01000000
/dev/mem opened.
Memory mapped at address 0xb6f53000.
Read at address 0x4804C190 (0xb6f53190):  0x01800000
Write at address 0x4804C190 (0xb6f53190): 0x01000000,
readback 0x01000000
```

- On

```
bone$ devmem2 0x4804c194 w 0x01000000
/dev/mem opened.
Memory mapped at address 0xb6f9f000.
Read at address 0x4804C194 (0xb6f9f194):  0x00800000
Write at address 0x4804C194 (0xb6f9f194): 0x01800000,
readback 0x01800000
```

# mmap()

- The same can be done more quickly from a C program using **`mmap()`**
- **`mmap()`** is a way of mapping a physical address space into a user-space program

```
volatile void *gpio_addr;
volatile unsigned int *gpio_setdataout_addr;
int fd = open("/dev/mem", O_RDWR);
gpio_addr = mmap(0, GPIO1_SIZE, PROT_READ | PROT_WRITE,
                    MAP_SHARED, fd, GPIO1_START_ADDR);
gpio_setdataout_addr = gpio_addr + GPIO_SETDATAOUT;
   0x4804c190          0x4804c000        0x190
*gpio_setdataout_addr = USR3;
                          (1<<24)
```

# Exercise GPIO via mmap

- Homework has you work through some examples
- **`gpioThru.c`** copies an input pin to an output