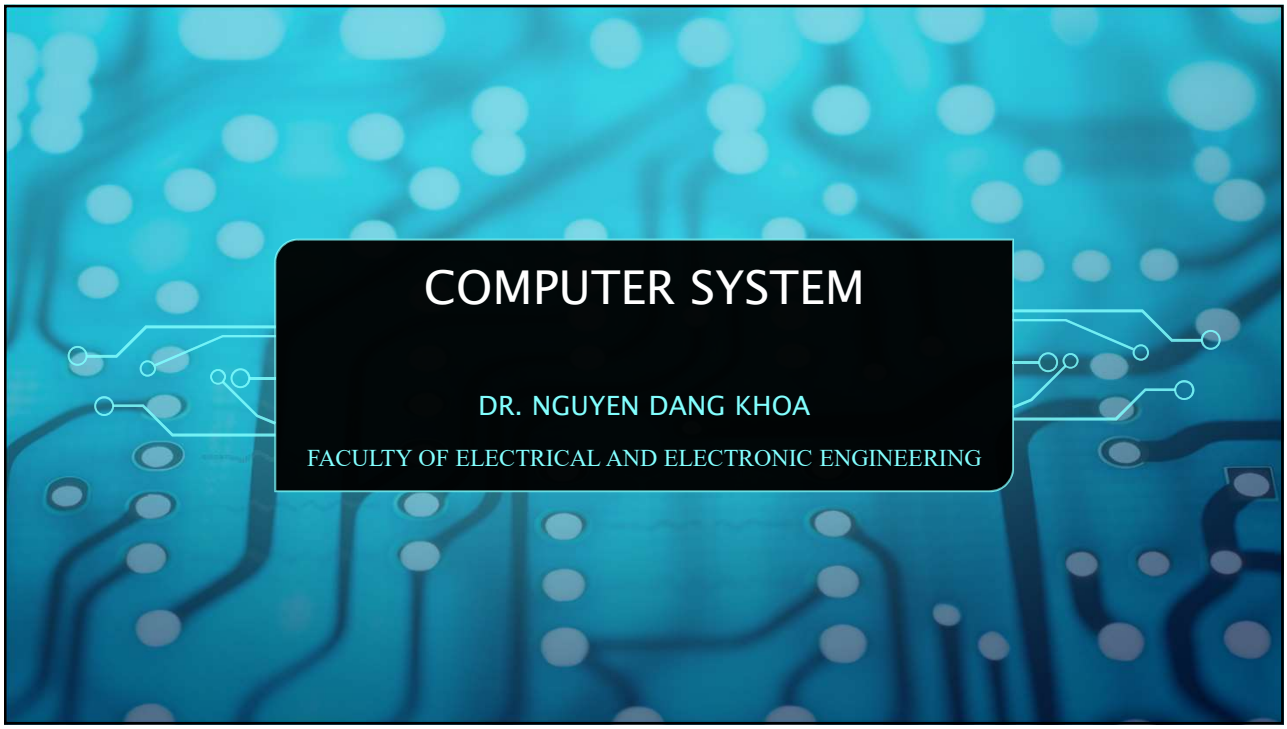
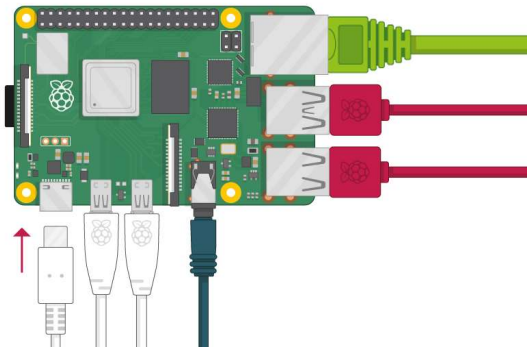



# ARM ASSEMBLY

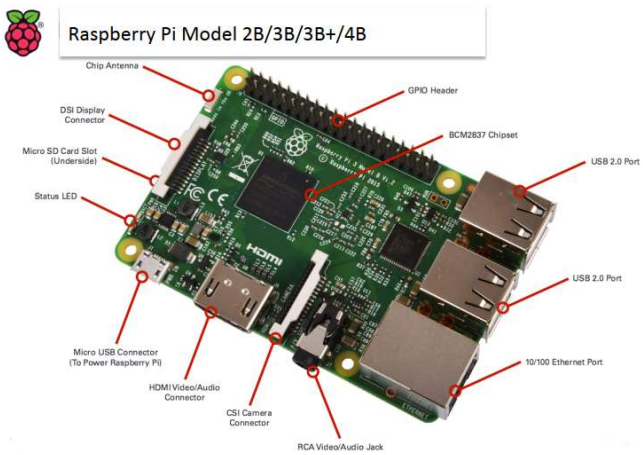
- **Material**
  1. Raspberry 3B+ https://hshop.vn/products/raspberry-pi-3-made-in-uk
  2. Led/breadboard/resistors/connector (50K) ➔ Arduino full https://shopee.vn/Combo-B%E1%BB%99-K%C3%ADt-Arduino-Uno-R3-Full-V3-2020-(BH-06-Th%C3%A1ng)-i.27117857.4103220867 (500K)
  3. HDMI Cable https://vhshopvn.com/product/day-hdmi-hdmi-14-10m-ugreen-10110/
  4. MicroSD 16G https://shopee.vn/Th%E1%BA%BB-nh%E1%BB%9B-Sandisk-MicroSD-Class-10-16GB-Ch%C3%ADnh-H%C3%A3ng-i.3558559.103483136
  5. USB Keyboard/Mouse (*optional*)
  6. Micro SD/SD and Micro USB convertor https://shopee.vn/%C4%90%E1%BA%A7u-%C4%91%E1%BB%8Dc-th%E1%BA%BB-Micro-SD-SD-chuy%E1%BB%83n-%C4%91%E1%BB%95i-t%E1%BB%AB-Micro-USB-OTG-sang-USB-2.0-i.64600333.1068104004
  7. FASM: Flat Assembler 1.4. https://arm.flatassembler.net/

# What do we study?



- Connect to monitor
- Connect MicroSD
- Connect GPIO
- Programming based on ASM code

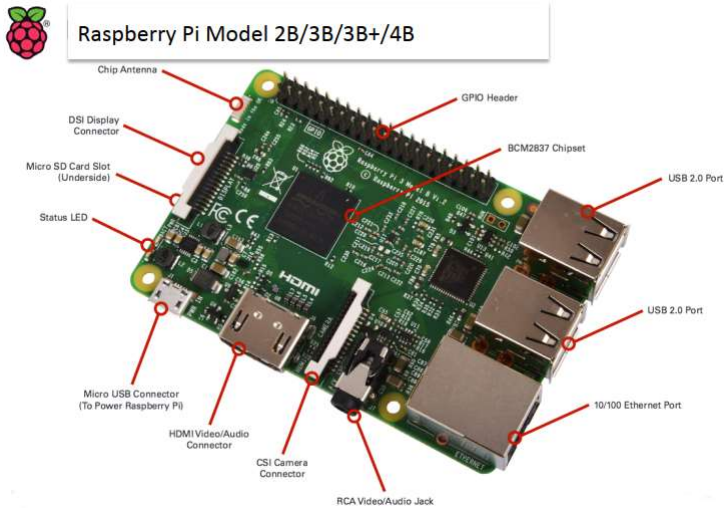
# What do we study?




Raspberry Pi 3 Model B uses CPU ARM Cortex-A53 Quadcore 1.2GHz 64-bit, RAM 1GB, Wifi 802.11n and Bluetooth 4.1
**Raspberry Pi 3:**

- Broadcom BCM2837 chipset running at 1.2 GHz
- 64-bit quad-core ARM Cortex-A53
- 802.11 b/g/n Wireless LAN
- Bluetooth 4.1 (Classic & Low Energy)
- Dual core Videocore IV® Multimedia co-processor
- 1 GB LPDDR2 memory
- Supports all the latest ARM GNU/Linux distributions and Windows 10 IoT
- MicroUSB connector for 2.5 A power supply
- 1 x 10/100 Ethernet port
- 1 x HDMI video/audio connector
- 1 x RCA video/audio connector
- 4 x USB 2.0 ports
- 40 GPIO pins
- Chip antenna
- DSI display connector
- MicroSD card slot
- Dimensions: 85 x 56 x 17 mm

# What do we study?

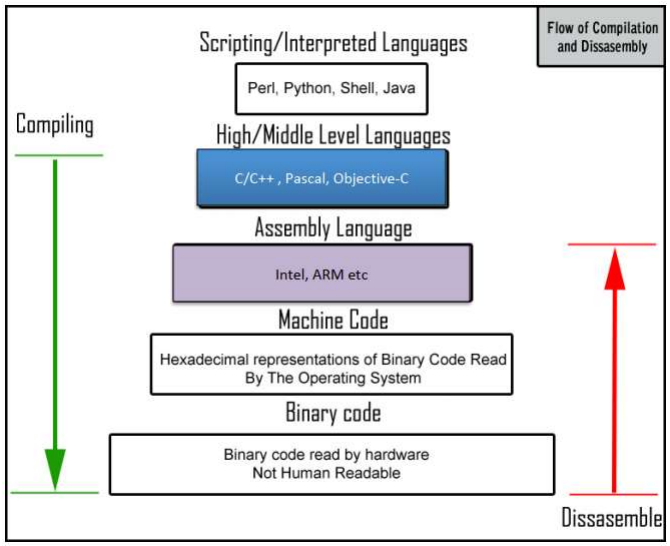



Raspberry Pi 3 GPIO Header

| Pin# | NAME | | NAME | Pin# |
|---|---|---|---|---|
| 01 | 3.3v DC Power | | DC Power 5v | 02 |
| 03 | GPIO02 (SDA1 , I²C) | | DC Power 5v | 04 |
| 05 | GPIO03 (SCL1 , I²C) | | Ground | 06 |
| 07 | GPIO04 (GPIO_GCLK) | | (TXD0) GPIO14 | 08 |
| 09 | Ground | | (RXD0) GPIO15 | 10 |
| 11 | GPIO17 (GPIO_GEN0) | | (GPIO_GEN1) GPIO18 | 12 |
| 13 | GPIO27 (GPIO_GEN2) | | Ground | 14 |
| 15 | GPIO22 (GPIO_GEN3) | | (GPIO_GEN4) GPIO23 | 16 |
| 17 | 3.3v DC Power | | (GPIO_GEN5) GPIO24 | 18 |
| 19 | GPIO10 (SPI_MOSI) | | Ground | 20 |
| 21 | GPIO09 (SPI_MISO) | | (GPIO_GEN6) GPIO25 | 22 |
| 23 | GPIO11 (SPI_CLK) | | (SPI_CE0_N) GPIO08 | 24 |
| 25 | Ground | | (SPI_CE1_N) GPIO07 | 26 |
| 27 | ID_SD (I²C ID EEPROM) | | (I²C ID EEPROM) ID_SC | 28 |
| 29 | GPIO05 | | Ground | 30 |
| 31 | GPIO06 | | GPIO12 | 32 |
| 33 | GPIO13 | | Ground | 34 |
| 35 | GPIO19 | | GPIO16 | 36 |
| 37 | GPIO26 | | GPIO20 | 38 |
| 39 | Ground | | GPIO21 | 40 |

Rev. 2 29/02/2016 www.element14.com/RaspberryPi

# Programming language




Why do you need to know it? (ASM)

– Write tiny/really fast programs

– No operating system to get in the way

– Write compilers

– Write drivers for custom hardware

– Find vulnerabilities in code.

# RPI ARM PROCESSORS

| Model | Processor | Instruction Set |
|---|---|---|
| Raspberry Pi 4 B  | Broadcom BCM2711, 1.5 GHz Quad core Cortex-A72 | ARMv8 64 bit |
| Raspberry Pi 3B  | Broadcom BCM2837 , 1.2 GHz Quad Core Cortex-A53 | ARMv8 64 bit |
| Raspberry Pi 2B  | Broadcom BCM2837, 900Mhz Quad Core Cortex-A7 | ARMv7 32 bit |

- ARMv8 supports 64 bit operations, but is also 32 bit compatible (RPi 3 & 4)
- Our unit uses with 32 bit instruction set

  ➔

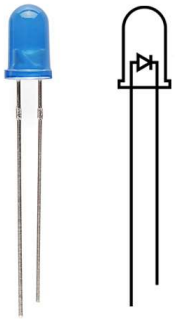
  - 32 bit-wide registers
  - 32 bit-wide addresses

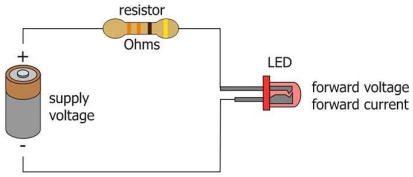
# How to install ASM code to Raspberry

1. Using FASMARM software for programming ➔ Compiler to *.BIN file
2. Rename *.BIN file to kernel.img/kernel7.img
3. Copy kernel.img/kernel7.img, start.elf, and bootcode.bin (find in the resources.zip) to microSD card
4. Put microSD card
5. Connect GPIO (*optional*)
6. Connect power

# LED control

1. Led







Assume that $U_{Led} = 2Volt \;\; and \;\; I_{Led} = 20mA$

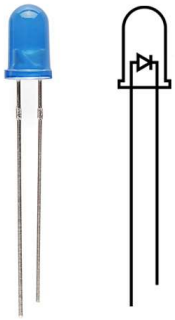
$$U_{Supply} = 9Volt$$

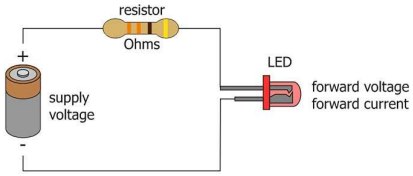
➔ R=?

https://www.youtube.com/watch?v=Rd9kvVs1ISQ

# LED control

1. Led







Assume that $U_{Led} = 2Volt \;\; and \;\; I_{Led} = 20mA$

$$U_{Supply} = 9Volt$$

$$RI + R_{Led} I = U_{Supply}$$

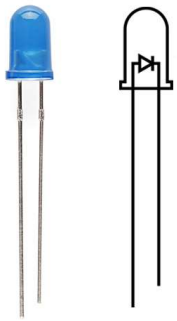
$$I = I_{led}$$

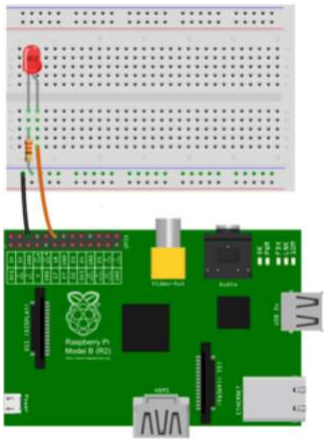
$$\Rightarrow R = \frac{U_{Supply} - R_{Led} I}{I} = \frac{U_{Supply} - U_{Led}}{I} = \frac{9-2}{20\left(10^{-3}\right)} = 350$$

*Select R=330R*

# LED control

2. Connection





Raspberry Pi 3 GPIO Header

| Pin# | NAME | | NAME | Pin# |
|---|---|---|---|---|
| 01 | 3.3v DC Power | | DC Power 5v | 02 |
| 03 | GPIO02 (SDA1 , I²C) | | DC Power 5v | 04 |
| 05 | GPIO03 (SCL1 , I²C) | | Ground | 06 |
| 07 | GPIO04 (GPIO_GCLK) | | (TXD0) GPIO14 | 08 |
| 09 | Ground | | (RXD0) GPIO15 | 10 |
| 11 | GPIO17 (GPIO_GEN0) | | (GPIO_GEN1) GPIO18 | 12 |
| 13 | GPIO27 (GPIO_GEN2) | | Ground | 14 |
| 15 | GPIO22 (GPIO_GEN3) | | (GPIO_GEN4) GPIO23 | 16 |
| 17 | 3.3v DC Power | | (GPIO_GEN5) GPIO24 | 18 |
| 19 | GPIO10 (SPI_MOSI) | | Ground | 20 |
| 21 | GPIO09 (SPI_MISO) | | (GPIO_GEN6) GPIO25 | 22 |
| 23 | GPIO11 (SPI_CLK) | | (SPI_CE0_N) GPIO08 | 24 |
| 25 | Ground | | (SPI_CE1_N) GPIO07 | 26 |
| 27 | ID_SD (I²C ID EEPROM) | | (I²C ID EEPROM) ID_SC | 28 |
| 29 | GPIO05 | | Ground | 30 |
| 31 | GPIO06 | | GPIO12 | 32 |
| 33 | GPIO13 | | Ground | 34 |
| 35 | GPIO19 | | GPIO16 | 36 |
| 37 | GPIO26 | | GPIO20 | 38 |
| 39 | Ground | | GPIO21 | 40 |

Rev. 2
29/02/2016

www.element14.com/RaspberryPi

https://www.youtube.com/watch?v=Rd9kvVs1lSQ

# ASM Programming

1. MOVE
2. ORR
3. LSL
4. STR
5. LDR
6. LOOP

# ASM Programming

1. MOVE

**mov r1, #20**

➔ r1=20 with r1 is register

*ARMv7 gives us 13 general purpose registers (**r0-r12**) to store values*

# ASM Programming

1. MOVE
2. ORR

**orr r1, $21**

➔ OR r1 with $21

OR could be used like ADD

**For example:**

```
BASE = $FE000000 ;
GPIO_OFFSET=$200000
mov     r0,BASE
orr     r0,GPIO_OFFSET
```

➔ R0= 0xFE200000

# ASM Programming

1. MOVE
2. ORR
3. LSL

**lsl r1, #21**

➔ **Logical shift left**

**For example:**

***mov r1,#1***

***lsl r1,#21***

➔ ***R1=1*** = 00000000 00000000 00000000 0000000**1**

➔ **Logical shift left** ➔ ***R1=*** 00000000 00**1**00000 00000000 00000000

# ASM Programming

1. MOVE
2. ORR
3. LSL
4. STR

**str r1, [r0, #16]**

➔ **Store value in R1 into memory R0 + offset 16**

**For example:**

*BASE = $3F000000 ;*

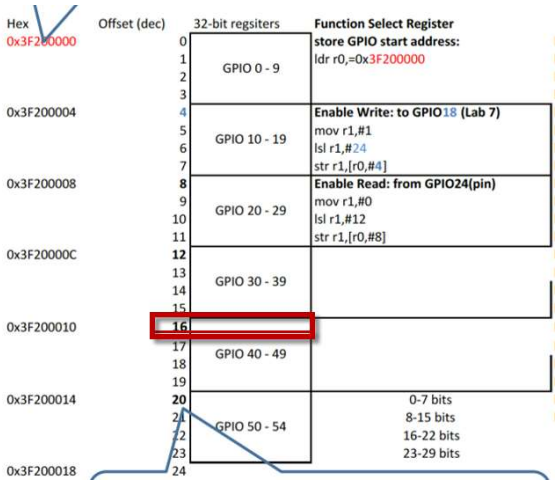
*GPIO_OFFSET=$200000*

*movr1, #20*

*mov r0,BASE*

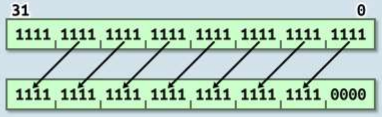
*orr r0,GPIO_OFFSET* ➔ R0= 0x3F200000

**str r1, [r0, #16]**



**LSL – Logical Shift Left**

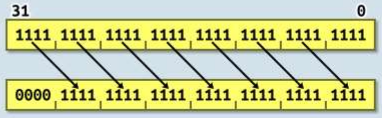
Example: Logical Shift Left by 4.




Equivalent to << in C.

**LSR – Logical Shift Right**

Example: Logical Shift Right by 4.




Equivalent to >> in C. i.e. unsigned division by a power of 2.

# ASM Programming

5. LDR

**ldr r0, [r1]**

➔ Load r0 with the value pointed to by r1

# ASM Programming

6. LOOP

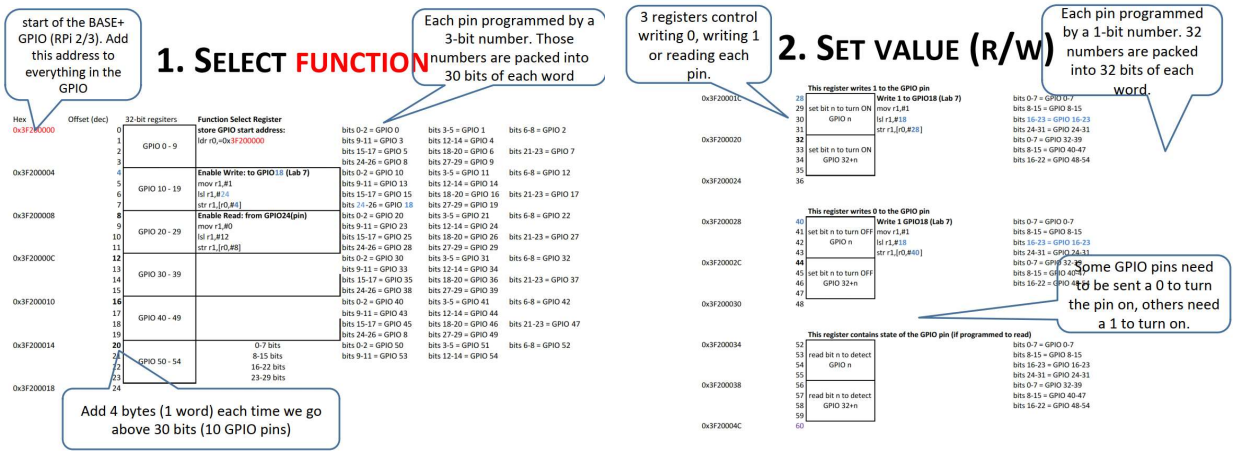
***Loop_function:***

***;;;; Work for looping***

***b Loop_function***

# Sample Programming: TURN ON/OFF LED

Map memory and register



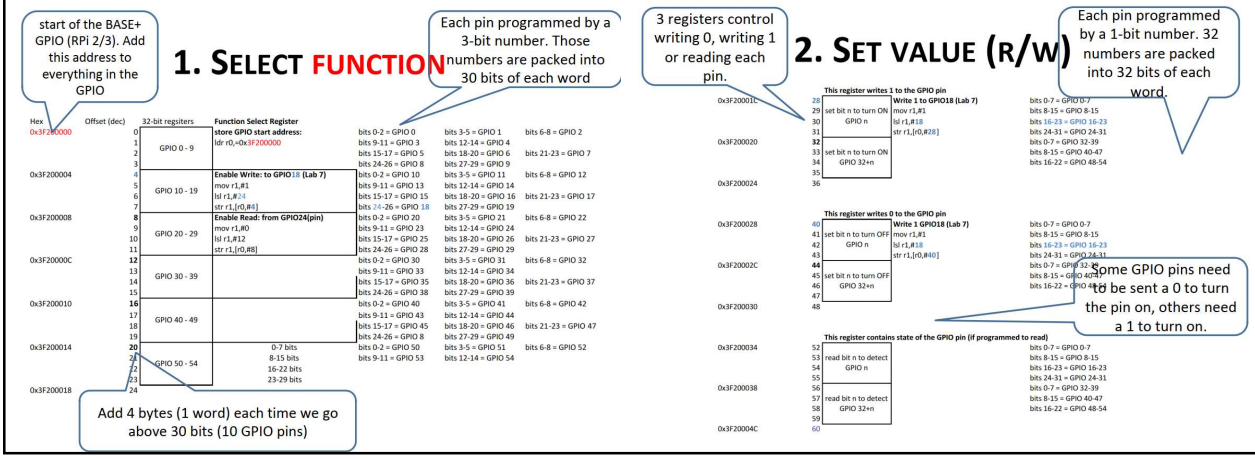
# Sample Programming: TURN ON/OFF LED

How to use?

Like other microchip Arduino, PIC, AVR... We need to have 2 steps

*Step 1: Configuration for GPIO in input/output mode* ➔ *1. Select function*

*Step 2: Set value ON/OFF for GPIO* ➔ *2. Set value (R/W)*

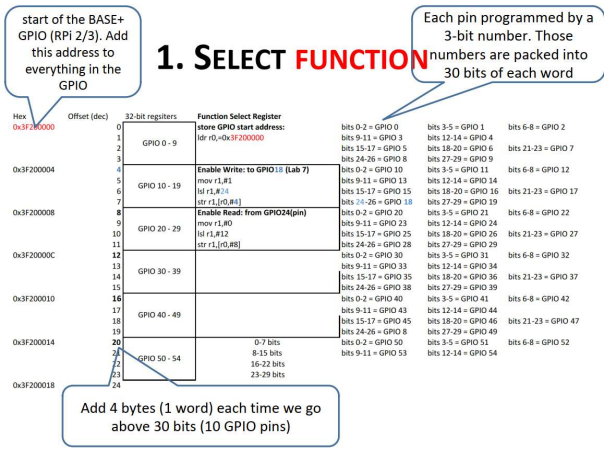



# Sample Programming: TURN ON/OFF LED

How to use?

Like other microchip Arduino, PIC, AVR... We need to have 2 steps

*Step 1: Configuration for GPIO in input/output mode* ➔ *1. Select function*




*From 0x3F200000 to 0x3F200018*

➔ *Present for GPIO0 to GPIO54*

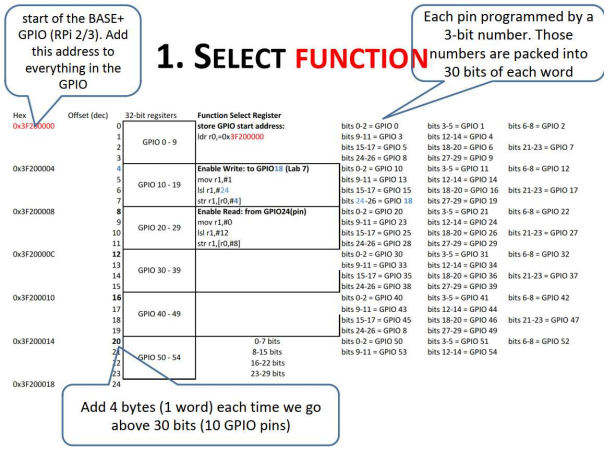
➔ *How to config?*

# Sample Programming: TURN ON/OFF LED

How to use?

Like other microchip Arduino, PIC, AVR... We need to have 2 steps

*Step 1: Configuration for GPIO in input/output mode* ➔ *1. Select function*




*For each GPIO, 3 bits to config is shown*

| Bit pattern | Pin Function |
|---|---|
| 000 | The pin in an input |
| 001 | The pin is an output |
| 100 | The pin does alternate function 0 |
| 101 | The pin does alternate function 1 |
| 110 | The pin does alternate function 2 |
| 111 | The pin does alternate function 3 |
| 011 | The pin does alternate function 4 |
| 010 | The pin does alternate function 5 |

*We will consider with 000 and 001*

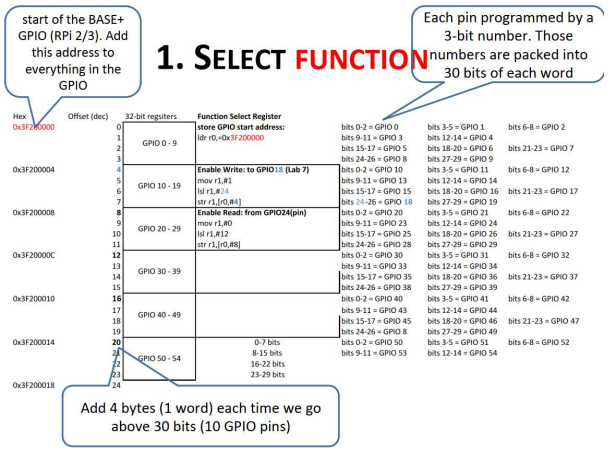
*You could see in the 1. Select function ➔ GPIO0 = bit 0-2 ➔ 3 bits...*

# Sample Programming: TURN ON/OFF LED

How to use?

Like other microchip Arduino, PIC, AVR... We need to have 2 steps

*Step 1: Configuration for GPIO in input/output mode* ➔ *1. Select function*




*For example: Defining GPIO35 is output*

- *GPIO35 is 0x3F20000C to 0x3F200010 (or offset 12)*

➔ *32 bit*

| 31 | 30 | .... | 17 | 16 | 15 | ... | ... | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|

- *GPIO35 is bit 15-17* ➔ *Set output configuration*

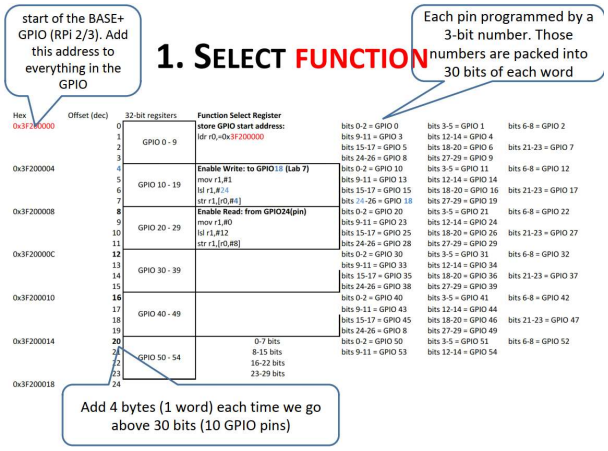
| 0 | 0 | 0 | 0 | 0 | 1 | ... | ... | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|

# Sample Programming: TURN ON/OFF LED

How to use?

Like other microchip Arduino, PIC, AVR... We need to have 2 steps

*Step 1: Configuration for GPIO in input/output mode* ➔ *1. Select function*




*For example: Defining GPIO35 is output*

- *GPIO35 is bit 15-17* ➔ *Set output configuration*

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | ... | ... | 0 | 0 |

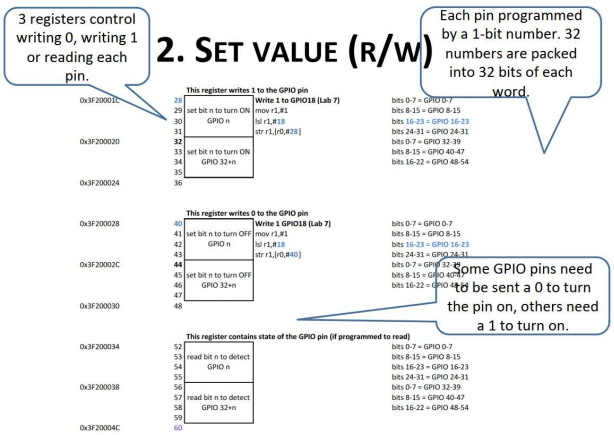
```
BASE     =     $3F000000
GPIO_OFFSET=$200000
mov     r0,BASE
orr     r0,GPIO_OFFSET
mov     r1,#1                ; R1=1
lsl       r1,#15             ;  shift 15 times
str       r1,[r0,#12]        ; [R0+12]=R1
```

# Sample Programming: TURN ON/OFF LED

How to use?

Like other microchip Arduino, PIC, AVR... We need to have 2 steps

*Step 2: Set value ON/OFF for GPIO* ➔ *2. Set value (R/W)*




*For example: Defining GPIO35 is output*

- *GPIO35 is 0x3F200020 to 0x3F200024 (or offset 32)*

*Bit 3th*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| GPIO39 | GPIO38 | GPIO37 | GPIO36 | GPIO35 | GPIO34 | GPIO33 | GPIO32 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

➔ *For set ON*

- *GPIO35 is 0x3F20002C to 0x3F200030 (or offset 44)*

*Bit 3th*

➔ *For set OFF*

# Sample Programming: TURN ON/OFF LED

How to use?

Like other microchip Arduino, PIC, AVR... We need to have 2 steps

*Step 2: Set value ON/OFF for GPIO* ➔ *2. Set value (R/W)*

```
mov    r1,#1          ; R1=1
lsl    r1,#3          ; 3 times
str    r1,[r0,#32]    ; [R0+32]=R1 ➔ ON

mov    r1,#1          ; R1=1
lsl    r1,#3          ; 3 times
str    r1,[r0,#44]    ; [R0+44]=R1 ➔ OFF
```

*For example: Defining GPIO35 is output*

- *GPIO35 is 0x3F200020 to 0x3F200024 (or offset 32)*

*Bit 3th*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| GPIO39 | GPIO38 | GPIO37 | GPIO36 | GPIO35 | GPIO34 | GPIO33 | GPIO32 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

➔ *For set ON*

- *GPIO35 is 0x3F20002C to 0x3F200030 (or offset 44)*

*Bit 3th*

➔ *For set OFF*

# Sample Programming: TURN ON/OFF LED

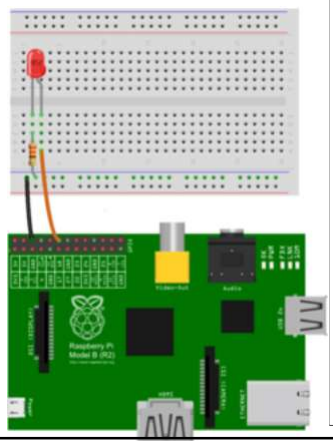
Full program for GPIO18

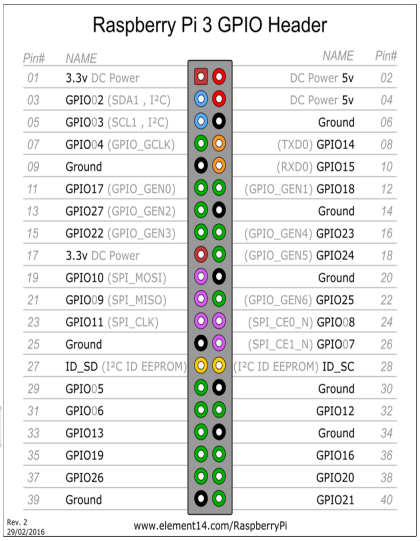
```
macro delay {

local .wait
mov r2,#0x3F0000

.wait:
    sub r2,#1
    cmp r2,#0
    bne .wait
}

BASE    =     $3F000000
GPIO_OFFSET=$200000
mov    r0,BASE
orr    r0,GPIO_OFFSET
mov    r1,#1
lsl    r1,#24
str    r1,[r0,#4]  ; finished select GPIO18

loop$:

mov    r1,#1
lsl    r1,#18
str    r1,[r0,#28]   ; 28=LED ON; 40=LED OFF
delay


mov    r1,#1
lsl    r1,#18
str    r1,[r0,#40]   ; 28=LED ON; 40=LED OFF
delay

b      loop$
```

1. Using FASMARM software for programming ➔ Compiler to *.BIN file
2. Rename *.BIN file to kernel.img/kernel7.img
3. Copy kernel.img/kernel7.img, start.elf, and bootcode.bin (find in the resources.zip) to microSD card
4. Put microSD card
5. Connect GPIO (*optional*)
6. Connect power

1. FASM: Flat Assembler 1.4.

https://arm.flatassembler.net/



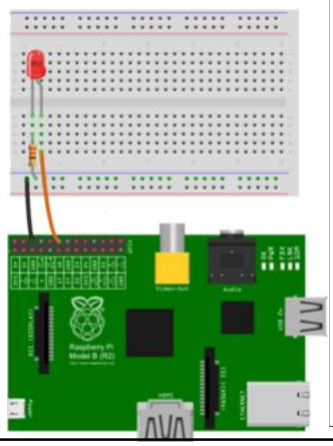


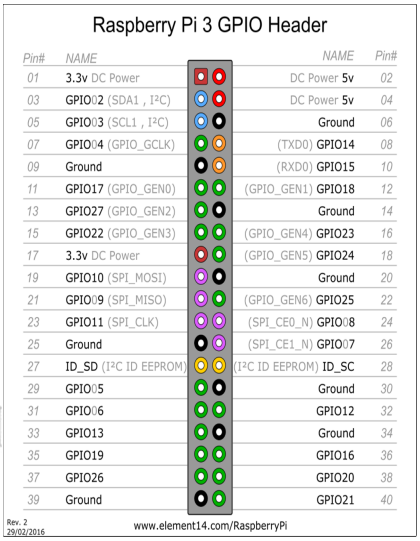

```
BASE    =   $3F000000
GPIO_OFFSET=$200000
mov     r0,BASE
orr     r0,GPIO_OFFSET

mov     r1,#1
lsl     r1,#24
str     r1,[r0,#4]  ; finished select GPIO18

mov     r1,#1
lsl     r1,#18
str     r1,[r0,#28]   ; 28=LED ON; 40=LED OFF


mov     r1,#1
lsl     r1,#18
str     r1,[r0,#40]   ; 28=LED ON; 40=LED OFF
```

1. Using FASMARM software for programming ➔ Compiler to *.BIN file
2. Rename *.BIN file to kernel.img/kernel7.img
3. Copy kernel.img/kernel7.img, start.elf, and bootcode.bin (find in the resources.zip) to microSD card
4. Put microSD card
5. Connect GPIO (*optional*)
6. Connect power

1. FASM: Flat Assembler 1.4.

https://arm.flatassembler.net/






```
macro delay{
local .wait
mov r2,#0x3F0000
.wait:
    sub r2,#1
    cmp r2,#0
    bne .wait
}

BASE    =   $3F000000
GPIO_OFFSET=$200000
mov     r0,BASE
orr     r0,GPIO_OFFSET
mov     r1,#1
lsl     r1,#24
str     r1,[r0,#4]  ; finished select GPIO18

loop$:
mov     r1,#1
lsl     r1,#18
str     r1,[r0,#28]   ; 28=LED ON; 40=LED OFF
delay

mov     r1,#1
lsl     r1,#18
str     r1,[r0,#40]   ; 28=LED ON; 40=LED OFF
delay

b       loop$
```