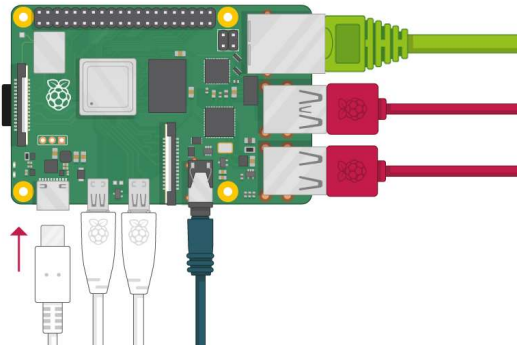


ARM ASSEMBLY

- **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](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?

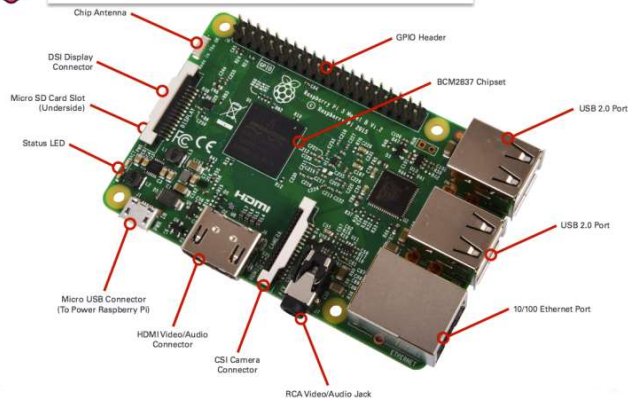


- Connect to monitor
- Connect MicroSD
- Connect GPIO
- Programming based on ASM code

What do we study?



Raspberry Pi Model 2B/3B/3B+/4B



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

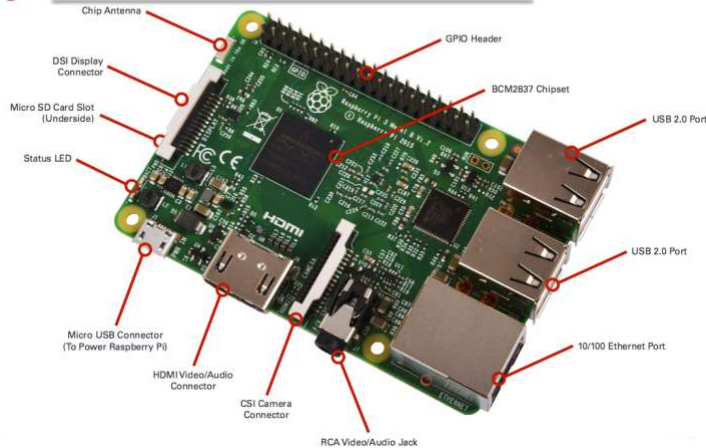
Raspberry Pi 3:

- 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 Model 2B/3B/3B+/4B



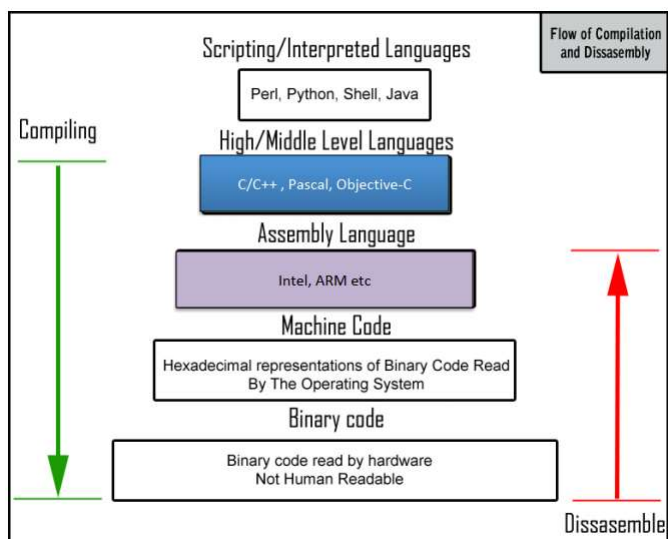
Raspberry Pi 3 GPIO Header

Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1, I ² C)		DC Power 5v	04
05	GPIO03 (SCL1, I ² C)		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 (I ² C ID EEPROM)		(I ² C 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



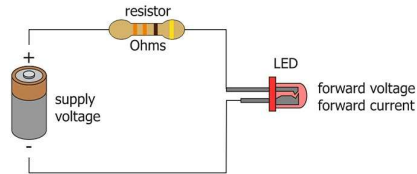
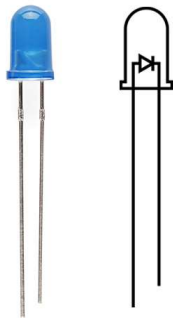
- 32 bit-wide registers
- 32 bit-wide addresses

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 $I_{Led} = 20mA$

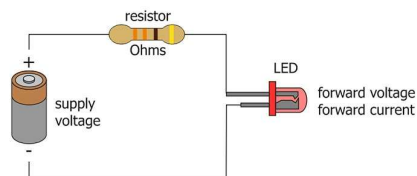
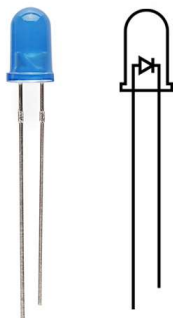
$$U_{Supply} = 9Volt$$

→ R=?

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

LED control

1. Led



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

$$U_{Supply} = 9Volt$$

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

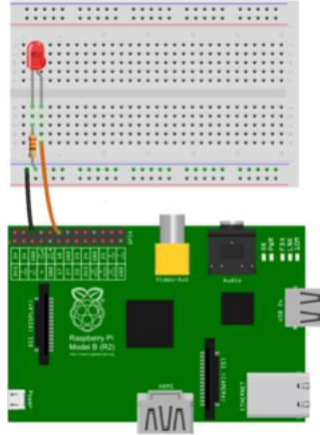
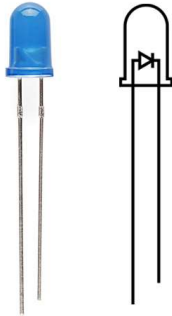
$$I = I_{led}$$

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

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 , I ² C)	DC Power 5v	04
05	GPIO03 (SCL1 , I ² C)	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 (I ² C ID EEPROM)	(I ² C 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=Rd9kvVs1ISQ>

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 ;

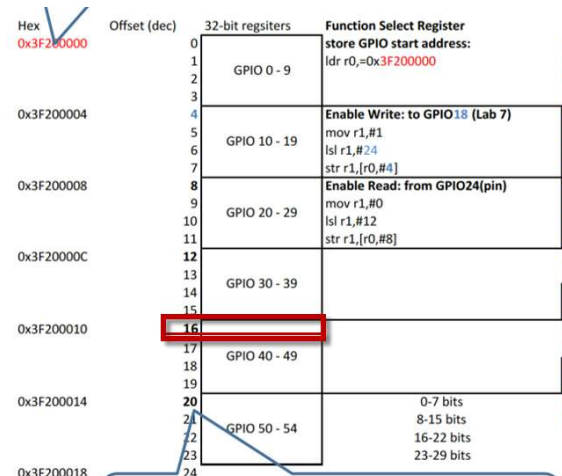
GPIO_OFFSET=\$200000

movr1, #20

mov r0, BASE

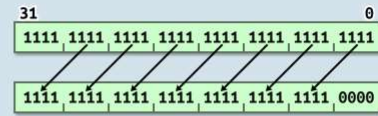
orr r0, GPIO_OFFSET → R0= 0x3F200000

str r1, [r0, #16]



LSL – Logical Shift Left

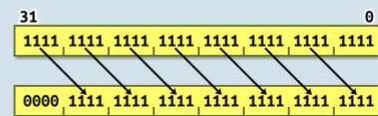
Example: Logical Shift Left by 4.



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

start of the BASE+
GPIO (RPI 2/3). Add
this address to
everything in the
GPIO

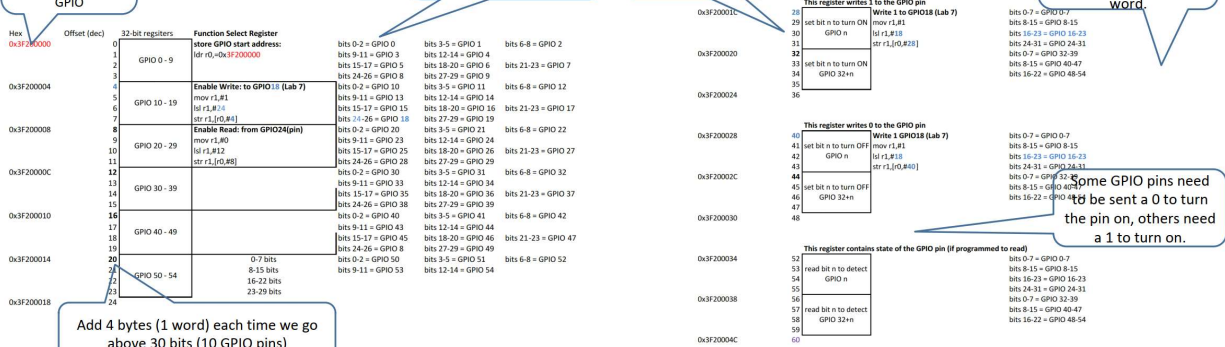
1. SELECT FUNCTION

Each pin programmed by a
3-bit number. Those
numbers are packed into
30 bits of each word.

3 registers control
writing 0, writing 1
or reading each
pin.

2. SET VALUE (R/W)

Each pin programmed
by a 1-bit number. 32
numbers are packed
into 32 bits of each
word.



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)

start of the BASE+ GPIO (RPI 2/3). Add this address to everything in the GPIO

1. SELECT FUNCTION

Each pin programmed by a 3-bit number. Those numbers are packed into 30 bits of each word

3 registers control writing 0, writing 1 or reading each pin.

2. SET VALUE (R/W)

Each pin programmed by a 1-bit number. 32 numbers are packed into 32 bits of each word.

Some GPIO pins need to be sent a 0 to turn the pin on, others need a 1 to turn on.

Add 4 bytes (1 word) each time we go above 30 bits (10 GPIO pins)

Hex	Offset (dec)	32-bit registers	Function Select Register	store GPIO start address:	bits 0-2 = GPIO 0	bits 3-5 = GPIO 1	bits 6-8 = GPIO 2
0x3F20000	0	GPIO 0 - 9	Enable Write to GPIO18 (Lab 7)	ldr r0, 0x3F200000	bits 9-11 = GPIO 3	bits 12-14 = GPIO 4	bits 15-17 = GPIO 5
0x3F200004	4	GPIO 10 - 19	Enable Read from GPIO24 (pin)	mov r1, #1 ldr r1, [r0, #4]	bits 18-20 = GPIO 6	bits 21-23 = GPIO 7	bits 24-26 = GPIO 8
0x3F200008	8	GPIO 20 - 29	mov r1, #0 ldr r1, [r0, #8]	bits 27-29 = GPIO 9	bits 3-5 = GPIO 10	bits 6-8 = GPIO 11	bits 9-11 = GPIO 12
0x3F20000C	12	GPIO 30 - 39	bits 12-14 = GPIO 13	bits 15-17 = GPIO 14	bits 18-20 = GPIO 15	bits 21-23 = GPIO 16	bits 24-26 = GPIO 17
0x3F200010	16	GPIO 40 - 49	bits 27-29 = GPIO 18	bits 3-5 = GPIO 19	bits 6-8 = GPIO 20	bits 9-11 = GPIO 21	bits 12-14 = GPIO 22
0x3F200014	20	GPIO 50 - 54	bits 15-17 = GPIO 23	bits 18-20 = GPIO 24	bits 21-23 = GPIO 25	bits 24-26 = GPIO 26	bits 27-29 = GPIO 27
0x3F200018	24		bits 2-4 = GPIO 28	bits 5-7 = GPIO 29	bits 8-10 = GPIO 30	bits 11-13 = GPIO 31	bits 14-16 = GPIO 32

This register writes 1 to the GPIO pin (Lab 7)

```

28: 0x3F200020: 1: Write 1 to GPIO18 (Lab 7)
29: set bit n to turn ON
30: GPIO n
31: ldr r1, #1
32: str r1, [r0, #28]
33: set bit n to turn ON
34: GPIO 32+n
35:
36:
37:
38:

```

This register writes 0 to the GPIO pin (Lab 7)

```

40: 0x3F200028: 0: Write 0 to GPIO18 (Lab 7)
41: set bit n to turn OFF
42: GPIO n
43: ldr r1, #0
44: str r1, [r0, #40]
45: set bit n to turn OFF
46: GPIO 32+n
47:
48:
49:
50:

```

This register contains state of the GPIO pin (if programmed to read)

```

52: 0x3F200034: 52: read bit n to detect
53: GPIO n
54:
55:
56:
57: read bit n to detect
58: GPIO 32+n
59:
60:

```

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

start of the BASE+ GPIO (RPI 2/3). Add this address to everything in the GPIO

1. SELECT FUNCTION

Each pin programmed by a 3-bit number. Those numbers are packed into 30 bits of each word

From 0x3F200000 to 0x3F200018
→ Present for GPIO0 to GPIO54

→ How to config?

Add 4 bytes (1 word) each time we go above 30 bits (10 GPIO pins)

Hex	Offset (dec)	32-bit registers	Function Select Register	store GPIO start address:	bits 0-2 = GPIO 0	bits 3-5 = GPIO 1	bits 6-8 = GPIO 2
0x3F20000	0	GPIO 0 - 9	Enable Write to GPIO18 (Lab 7)	ldr r0, 0x3F200000	bits 9-11 = GPIO 3	bits 12-14 = GPIO 4	bits 15-17 = GPIO 5
0x3F200004	4	GPIO 10 - 19	Enable Read from GPIO24 (pin)	mov r1, #1 ldr r1, [r0, #4]	bits 18-20 = GPIO 6	bits 21-23 = GPIO 7	bits 24-26 = GPIO 8
0x3F200008	8	GPIO 20 - 29	mov r1, #0 ldr r1, [r0, #8]	bits 27-29 = GPIO 9	bits 3-5 = GPIO 10	bits 6-8 = GPIO 11	bits 9-11 = GPIO 12
0x3F20000C	12	GPIO 30 - 39	bits 12-14 = GPIO 13	bits 15-17 = GPIO 14	bits 18-20 = GPIO 15	bits 21-23 = GPIO 16	bits 24-26 = GPIO 17
0x3F200010	16	GPIO 40 - 49	bits 27-29 = GPIO 18	bits 3-5 = GPIO 19	bits 6-8 = GPIO 20	bits 9-11 = GPIO 21	bits 12-14 = GPIO 22
0x3F200014	20	GPIO 50 - 54	bits 15-17 = GPIO 23	bits 18-20 = GPIO 24	bits 21-23 = GPIO 25	bits 24-26 = GPIO 26	bits 27-29 = GPIO 27
0x3F200018	24		bits 2-4 = GPIO 28	bits 5-7 = GPIO 29	bits 8-10 = GPIO 30	bits 11-13 = GPIO 31	bits 14-16 = GPIO 32

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

start of the BASE+ GPIO (RPI 2/3). Add this address to everything in the GPIO

1. SELECT FUNCTION

Each pin programmed by a 3-bit number. Those numbers are packed into 30 bits of each word

For each GPIO, 3 bits to config is shown

Bit pattern	Pin Function
000	The pin is 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

Add 4 bytes (1 word) each time we go above 30 bits (10 GPIO pins)

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

start of the BASE+ GPIO (RPI 2/3). Add this address to everything in the GPIO

1. SELECT FUNCTION

Each pin programmed by a 3-bit number. Those numbers are packed into 30 bits of each word

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
---	---	---	---	---	---	-----	-----	---	---

Add 4 bytes (1 word) each time we go above 30 bits (10 GPIO pins)

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

start of the BASE+GPIO (RPI 2/3). Add this address to everything in the GPIO

1. SELECT FUNCTION

Each pin programmed by a 3-bit number. Those numbers are packed into 30 bits of each word

Add 4 bytes (1 word) each time we go above 30 bits (10 GPIO pins)

Hex	Offset (dec)	32-bit registers	Function Select Register
0x3F20000	0	GPIO 0 - 9	store GPIO start address: ldr r0,=0x3F20000
0x3F20004	4	GPIO 10 - 19	Enable Write to GPIO18 (Lab 7) mov r1,#1 ldr r1,[r0,8] str r1,[r0,8]
0x3F20008	8	GPIO 20 - 29	Enable Read from GPIO24(pin) mov r1,#0 ldr r1,[r0,12] str r1,[r0,8]
0x3F2000C	12	GPIO 30 - 39	
0x3F20010	16	GPIO 40 - 49	
0x3F20014	20	GPIO 50 - 54	0-7 bits 8-15 bits 16-23 bits 24-31 bits
0x3F20018	24		

bits 0-2 = GPIO 0
bits 3-5 = GPIO 1
bits 6-8 = GPIO 2
bits 9-11 = GPIO 3
bits 12-14 = GPIO 4
bits 15-17 = GPIO 5
bits 18-20 = GPIO 6
bits 21-23 = GPIO 7
bits 24-26 = GPIO 8
bits 27-29 = GPIO 9
bits 30-32 = GPIO 10
bits 33-35 = GPIO 11
bits 36-38 = GPIO 12
bits 39-41 = GPIO 13
bits 42-44 = GPIO 14
bits 45-47 = GPIO 15
bits 48-50 = GPIO 16
bits 51-53 = GPIO 17
bits 54-56 = GPIO 18
bits 57-59 = GPIO 19
bits 60-62 = GPIO 20
bits 63-65 = GPIO 21
bits 66-68 = GPIO 22
bits 69-71 = GPIO 23
bits 72-74 = GPIO 24
bits 75-77 = GPIO 25
bits 78-80 = GPIO 26
bits 81-83 = GPIO 27
bits 84-86 = GPIO 28
bits 87-89 = GPIO 29
bits 90-92 = GPIO 30
bits 93-95 = GPIO 31
bits 96-98 = GPIO 32
bits 99-101 = GPIO 33
bits 102-104 = GPIO 34
bits 105-107 = GPIO 35
bits 108-110 = GPIO 36
bits 111-113 = GPIO 37
bits 114-116 = GPIO 38
bits 117-119 = GPIO 39
bits 120-122 = GPIO 40
bits 123-125 = GPIO 41
bits 126-128 = GPIO 42
bits 129-131 = GPIO 43
bits 132-134 = GPIO 44
bits 135-137 = GPIO 45
bits 138-140 = GPIO 46
bits 141-143 = GPIO 47
bits 144-146 = GPIO 48
bits 147-149 = GPIO 49
bits 150-152 = GPIO 50
bits 153-155 = GPIO 51
bits 156-158 = GPIO 52
bits 159-161 = GPIO 53
bits 162-164 = GPIO 54

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)

3 registers control writing 0, writing 1 or reading each pin.

2. SET VALUE (R/W)

Each pin programmed by a 1-bit number. 32 numbers are packed into 32 bits of each word.

Some GPIO pins need to be sent a 0 to turn the pin on, others need a 1 to turn on.

Hex	Offset (dec)	32-bit registers	Function Select Register
0x3F2001C	28	GPIO 0 - 9	Write 1 to GPIO18 (Lab 7) mov r1,#1 ldr r1,[r0,8] str r1,[r0,8]
0x3F20020	32	GPIO 10 - 19	
0x3F20024	36	GPIO 20 - 29	
0x3F20028	40	GPIO 30 - 39	Write 0 to the GPIO pin mov r1,#0 ldr r1,[r0,8] str r1,[r0,8]
0x3F2002C	44	GPIO 40 - 49	
0x3F20030	48	GPIO 50 - 54	
0x3F20034	52	GPIO 55 - 59	
0x3F20038	56	GPIO 60 - 64	
0x3F2003C	60	GPIO 65 - 69	
0x3F20040	64	GPIO 70 - 74	

bits 0-7 = GPIO 0-7
bits 8-15 = GPIO 8-15
bits 16-23 = GPIO 16-23
bits 24-31 = GPIO 24-31
bits 32-39 = GPIO 32-39
bits 40-47 = GPIO 40-47
bits 48-55 = GPIO 48-55
bits 56-63 = GPIO 56-63
bits 64-71 = GPIO 64-71
bits 72-79 = GPIO 72-79
bits 80-87 = GPIO 80-87
bits 88-95 = GPIO 88-95
bits 96-103 = GPIO 96-103
bits 104-111 = GPIO 104-111
bits 112-119 = GPIO 112-119
bits 120-127 = GPIO 120-127
bits 128-135 = GPIO 128-135
bits 136-143 = GPIO 136-143
bits 144-151 = GPIO 144-151
bits 152-159 = GPIO 152-159
bits 160-167 = GPIO 160-167
bits 168-175 = GPIO 168-175
bits 176-183 = GPIO 176-183
bits 184-191 = GPIO 184-191
bits 192-199 = GPIO 192-199
bits 200-207 = GPIO 200-207
bits 208-215 = GPIO 208-215
bits 216-223 = GPIO 216-223
bits 224-231 = GPIO 224-231
bits 232-239 = GPIO 232-239
bits 240-247 = GPIO 240-247
bits 248-255 = GPIO 248-255
bits 256-263 = GPIO 256-263
bits 264-271 = GPIO 264-271
bits 272-279 = GPIO 272-279
bits 280-287 = GPIO 280-287
bits 288-295 = GPIO 288-295
bits 296-303 = GPIO 296-303
bits 304-311 = GPIO 304-311
bits 312-319 = GPIO 312-319
bits 320-327 = GPIO 320-327
bits 328-335 = GPIO 328-335
bits 336-343 = GPIO 336-343
bits 344-351 = GPIO 344-351
bits 352-359 = GPIO 352-359
bits 360-367 = GPIO 360-367
bits 368-375 = GPIO 368-375
bits 376-383 = GPIO 376-383
bits 384-391 = GPIO 384-391
bits 392-399 = GPIO 392-399
bits 400-407 = GPIO 400-407
bits 408-415 = GPIO 408-415
bits 416-423 = GPIO 416-423
bits 424-431 = GPIO 424-431
bits 432-439 = GPIO 432-439
bits 440-447 = GPIO 440-447
bits 448-455 = GPIO 448-455
bits 456-463 = GPIO 456-463
bits 464-471 = GPIO 464-471
bits 472-479 = GPIO 472-479
bits 480-487 = GPIO 480-487
bits 488-495 = GPIO 488-495
bits 496-503 = GPIO 496-503
bits 504-511 = GPIO 504-511
bits 512-519 = GPIO 512-519
bits 520-527 = GPIO 520-527
bits 528-535 = GPIO 528-535
bits 536-543 = GPIO 536-543
bits 544-551 = GPIO 544-551
bits 552-559 = GPIO 552-559
bits 560-567 = GPIO 560-567
bits 568-575 = GPIO 568-575
bits 576-583 = GPIO 576-583
bits 584-591 = GPIO 584-591
bits 592-599 = GPIO 592-599
bits 600-607 = GPIO 600-607
bits 608-615 = GPIO 608-615
bits 616-623 = GPIO 616-623
bits 624-631 = GPIO 624-631
bits 632-639 = GPIO 632-639
bits 640-647 = GPIO 640-647
bits 648-655 = GPIO 648-655
bits 656-663 = GPIO 656-663
bits 664-671 = GPIO 664-671
bits 672-679 = GPIO 672-679
bits 680-687 = GPIO 680-687
bits 688-695 = GPIO 688-695
bits 696-703 = GPIO 696-703
bits 704-711 = GPIO 704-711
bits 712-719 = GPIO 712-719
bits 720-727 = GPIO 720-727
bits 728-735 = GPIO 728-735
bits 736-743 = GPIO 736-743
bits 744-751 = GPIO 744-751
bits 752-759 = GPIO 752-759
bits 760-767 = GPIO 760-767
bits 768-775 = GPIO 768-775
bits 776-783 = GPIO 776-783
bits 784-791 = GPIO 784-791
bits 792-799 = GPIO 792-799
bits 800-807 = GPIO 800-807
bits 808-815 = GPIO 808-815
bits 816-823 = GPIO 816-823
bits 824-831 = GPIO 824-831
bits 832-839 = GPIO 832-839
bits 840-847 = GPIO 840-847
bits 848-855 = GPIO 848-855
bits 856-863 = GPIO 856-863
bits 864-871 = GPIO 864-871
bits 872-879 = GPIO 872-879
bits 880-887 = GPIO 880-887
bits 888-895 = GPIO 888-895
bits 896-903 = GPIO 896-903
bits 904-911 = GPIO 904-911
bits 912-919 = GPIO 912-919
bits 920-927 = GPIO 920-927
bits 928-935 = GPIO 928-935
bits 936-943 = GPIO 936-943
bits 944-951 = GPIO 944-951
bits 952-959 = GPIO 952-959
bits 960-967 = GPIO 960-967
bits 968-975 = GPIO 968-975
bits 976-983 = GPIO 976-983
bits 984-991 = GPIO 984-991
bits 992-999 = GPIO 992-999

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

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
}

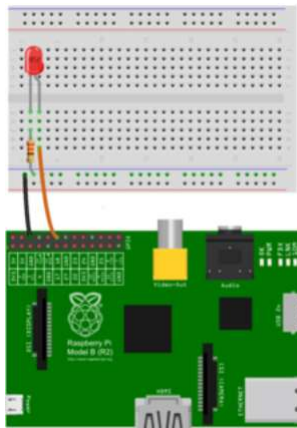
BASE = $3F000000
GPIO_OFFSET=$200000
mov r0,BASE
orr r0,GPIO_OFFSET
mov r1,#1
lsl r1,#24
str r1,[r0,#4] ; finished select GPIO18

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



Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1, I ² C)		DC Power 5v	04
05	GPIO03 (SCL1, I ² C)		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 (I ² C ID EEPROM)		(I ² C 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

```

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

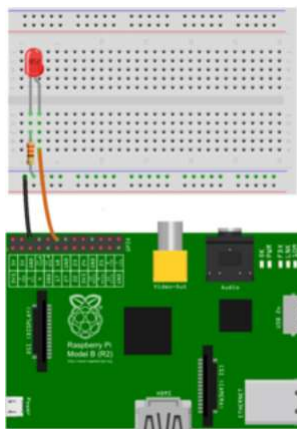
```

```

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



Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1, I ² C)		DC Power 5v	04
05	GPIO03 (SCL1, I ² C)		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 (I ² C ID EEPROM)		(I ² C 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

```

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$

```