

ARM Assembly

ARM Assembly

by [Shadow Player](#) on 07 Jan 2009 18:24

Hi SE Developers!

In this topic we will post ASM instructions.
Please post as this:

ADD

- Meaning: Arithmetic Addition
- Instruction Set: THUMB
- Other mnemonics: ADDS
- Definition: Mathematically add two registers or a register and a number not greater than 0xFF (255).
- Usage:

Code: [Select all](#)

```
ADD R1, R2, 1
```

$R1 = R2 + 1$

Code: [Select all](#)

```
ADD R1, R2, R3
```

$R1 = R2 + R3$

Code: [Select all](#)

```
ADD R1, 2
```

$R1 = R1 + 2$

Regards!

RE: ARM Assembly

by [Shadow Player](#) on 07 Jan 2009 18:33

B

- Meaning: Branch
- Instruction Set: THUMB
- Other mnemonics: none

- Definition: Jumps execution to a new offset. It has a maximum reach, +/- 2MB.
- Usage:

Code: [Select all](#)

```
B 0x45123456
```

Jumps execution to address 0x45123456.

- Note: Destination address has to be in reach and aligned.
-

RE: ARM Assembly

by [Shadow Player](#) on 11 Jan 2009 19:27

LDR

- Meaning: Load Register
- Instruction Set: THUMB
- Other Mnemonics: none
- Definition: Loads a register with a chunk of data. Default is 1 dword (4 bytes)
- Variations: LDRB (Loads 1 byte) LDRH (Loads 2 bytes)
- Usage:

Code: [Select all](#)

```
LDR R1, dword_45123456
...
align 4
dword_45123456 DCD 0xBABE
```

Loads 0xBABE into R1

Code: [Select all](#)

```
LDR R1, [R2]
```

Loads data stored at the address contained in R2 to R1

- Note: Destination address has to be in reach and aligned.
-

RE: ARM Assembly

by [Vitor Boss®](#) on 16 Feb 2010 16:56

CMP

- Meaning: Compare Register
- Instruction Set: THUMB
- Other Mnemonics: none
- Definition: CMP allows you to compare the contents of a register with another register or an immediate value, updating the status flags to allow conditional execution to take place. It performs a subtraction, but does not store the result in status register.
- Variations: unknown
- Usage:
CMP
same of:
status = op_1 - op_2

Code: [Select all](#)

```
CMP R0, #0xA  
BHE 121407A5; See next
```

- Note: Destination address has to be in reach and aligned.[hr][hr]

BHE

- Meaning: Branch if Higher or Equal
- Instruction Set: THUMB
- Other Mnemonics: none
- Definition: Use the Status register and a CMP instruction internally as an "IF" function. Always is used after a CMP instruction.
- Variations: BEQ; BNE; BHI and BLE
- Usage:

Code: [Select all](#)

```
CMP R0, #0xA; Status = R0 - 0xA  
BHE 121407A5; If Status = 0 Branch 121407A5
```

by [Shadow Player](#) on 16 Feb 2010 19:05

Other Related:

Code: [Select all](#)

```
BEQ = Branch if Equal
BNE = Branch if Not Equal
BHI = Branch if Higher
BLE = Branch if Less or Equal
```

[RE: Ask for the plugin for IDA](#)

by [Mojsa](#) on 24 Sep 2009 19:22

BNE is used in:

```
cmp rx,x ;compare an register for an value
bne _exitorsomething ;if it's not value written in cmp instruction,it'll branch to _exitorsomething
```

[RE: ARM Assembly](#)

by [Vitor Boss@](#) on 17 Feb 2010 00:58

Thank you.

STR

- Meaning: Store Register
- Instruction Set: THUMB
- Other Mnemonics: none
- Definition: These instructions load and store the value of Rx to the specified address in memory, is similar the LDR.
- Variations: STRB (Loads 1 byte) STRH (Loads 2 bytes)
- Usage:
Â STR

Code: [Select all](#)

```
LDR R1, dword_x
STR R0, [R1]
align 4
dword_x DCD 0x2A2AC3EC
```

- Note: Destination address must be a register or a Heap address.
-

RE: ARM Assembly

by [Shadow Player](#) on 18 Feb 2010 11:25

NOP (SE opcode bytes: C046)

- Meaning: No Operation
- Instruction Set: THUMB
- Other Mnemonics: None
- Definition: Does nothing.
- Usage:

Code: [Select all](#)

```
...  
NOP ;will do nothing, just continue.  
...
```

e.g. replace BL (branch&link) with NOP to stop it from being executed

RE: ARM Assembly

by [Shadow Player](#) on 18 Feb 2010 17:54

Labels & Entrypoint Names in Assembly

Many people have ported patches for years and understand most of Assembly but have never understood the labeling/names inside an ASM file, here is how it goes.

Example Code:

Code: [Select all](#)

```
LDR R1, dword_45123456  
B [R1]  
align 4  
dword_45123456 DCD 0x45123400
```

Q: What is **dword_45123456**?

A: It's a **label**

Q: What's a label good for?

A: To make **position references** inside the code, to **organize** structure and **avoid remembering offsets** in a single peace of code.

Q: Do I have to name it like 'loc_xxx', 'unk_xxx', 'sub_xxx' like IDA names them?

A: No. You can name them how ever you like, just make sure it's a single word (no white spaces).

New Example:

Code: [Select all](#)

```
B    foo_bar
...
foo_bar:
LDR  R1, cheese_burger
BLX  R1
align 4
cheese_burger DCD 0x45123400+1
```

In this example we have two labels: **foo_bar** and **cheese_burger**.

Now, another example with a branch reaching out of the code boundaries:

Code: [Select all](#)

```
B    rab_1
...
rab_1:
...
B    0x45809070
```

rab_1 is a label defined in our assembly file, therefrom we can jump to it very simple by branching to it.

However, the offset **0x45809070** is **NOT** inside our code boundaries. So we've got to write the address manually.

I hope this helped somebody to understand how it works.

Regards

RE: ARM Assembly

by [jamesbond22](#) on 20 Feb 2010 19:26

Code: [Select all](#)

```
LDR R2, _0x4527AF0E ;R2 = ???????? ?? ?????? (PC + 44)=[0x01A8ECE0]=
0x4527AF0E
MOV R3, R4 ;R3 = R4 = 0x2A (42)
MOV R1, 0x7D ;R1 = 125 "}"
LSL R1, R1, 3 ;R1 = R1 << 3 = 0x3E8 (1000)
MOV R0, R7 ;R0 = R7
PUSH {R0-R7,LR} ;????????? ???????? R0-R7,LR
MOV R0, 0x0 ;R0 = 0
```

```

SUB SP, SP, 0x4 ;SP = SP - 4
STR R0, [SP] ;????? ?? ?????? [SP] = R0
SUB SP, SP, 0x88 ;SP = SP - 136
MOV R0, SP ;R0= SP
LDR R1, _0x4529D5D9 ;R1 = ???????? ?? ?????? (PC + 28)=[0x01A8ECE4]=
0x4529D5D9
BLX R1 ;??????? ?? ?????? ? R1
MOV R1, 0x7D ;R1 = 125 "}"
LSL R1, R1, 3 ;R1 = R1 << 3 = 0x3E8 (1000)
MOV R2, 0x32 ;R2 = 50 "2"
MOV R3, R1 ;R3 = R1 = 0x3E8 (1000)
LDR R6, [R0] ;R6 = ?????? ?? ?????? [R0]
MOV R4, 0xFF ;R4 = 255 "Ë™"
ADD R4, 0x69 ;R4 = R4 + 105 "i" = 0x168 (360)
LDR R6, [R6, R4] ;R6 = ?????? ?? ?????? [R6 + R4]
BLX R6 ;??????? ?? ?????? ? R6
ADD SP, SP, 0x8C ;SP = SP + 140
POP {R0-R7,PC} ;????????? ?????????? R0-R7,PC
_0x4527AF0E DCD 0x453DF895+1 ;??????? ??? ???????? ?? ?????? 0x1A8ECB0
_0x4529D5D9 DCD 0x4540219C+1

```

RE: ARM Assembly

by [Shadow Player](#) on 20 Feb 2010 22:54

I found something great for all you IDA lovers: a **THUMB Decompiler**

<http://www.openrce.org/downloads/detail ... Decompiler>

by [jamesbond22](#) on 21 Feb 2010 16:58

The above plugin is for version v.4.7 only

IDA v4.7 download:

http://www.4shared.com/file/227287568/c ... v_47.html

RE: ARM Assembly

by [Vitor Boss®](#) on 22 Feb 2010 22:32

<http://www.heyrick.co.uk/assembler/qfinder.html>

Enjoy :kool: