LED

5. LED

5.1 ARM

IMX6UL Cortex-A7 Cortex-A7

:

- DEN0013D_cortex_a_series_PG.pdf
- ARM® Cortex™-A Series Version: 4.0 Programmer’s Guide.pdf
- : 00_UserManual\ \Arm \ARMv7 (DEN0013D_cortex_a_series_PG).pdf
- : 3: ARM Processor Modes and Registers

5.1.1

Cortex-A7 9 User Sys(System) FIQ IRQ ABT(Abort) SVC(Supervisor) UND(Undef) MON(

User	
Sys(System)	
FIQ	FIQ
IRQ	
ABT(Abort)	
SVC(Supervisor)	
UND(Undef)	
MON(Monitor)	
Hyp	

User 8

5.1.2

Cortex-A7

Cortex-A7 9

User (R0~R7)

FIQ R13(SP) SP_fiq

SVC R13(SP) SP_svc

9 34

1. R0~R7
2. R8~R14
3. R15
4.

4

5.1.2.1

R0~R7 R0~R7 R0~R7 R0~R7

5.1.2.2

R8~R12 (FIQ) Rx_irq(x=8~12) Rx(8~12) FIQ

R13(SP) 8 User Sys 7 7

R14(LR) 7 User Sys Hyp 6 6

R14(LR) BL BLX R14(LR) R14(LR) R15(PC)

5.1.2.3

R15(PC) 8

ARM -> -> R15(PC) R15(PC)

R15(PC) = + 8

5.1.2.4

PSR CPSR SPSR

CPSR CPSR CF

SPSR CPSR SPSR CPSR

N(bit31) ( ) N N=1/0 /

Z(bit30) CMP Z=1

C(bit29)

C=1 C=0

C=0 C=1

/ C

/ C

V(bit28) / V=1 V

Q(bit27) ARM v5TE_J Q=1/0 /

IT1:0 IT7:2 IT[7:0] IF-THEN

J(bit24) T(bit5) ARM Thumb

J	T	
0	0	ARM
0	1	Thumb
1	1	ThumbEE
1	0	Jazelle

GE3:0 SIMD

E(bit9) E=1/0 /

A(bit8) A=1

I(bit7) I=1/0 / IRQ

F(bit6) F=1/0 / FIQ

M[4:0]

M[4:0]	
10000	User
10001	FIQ
10010	IRQ
10011	Supervisor(SVC)
10110	Monitor(MON)
10111	Abort(ABT)
11010	Hyp(HYP)
11011	Undef(UND)
11111	System(SYS)

5.2

:

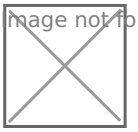
- DDI0406C_d_armv7ar_arm.pdf
- ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition
- : 00_UserManual\ \Arm \ armv7 ar CPU DDI0406C_d_armv
- : A5: ARM Instruction Set Encoding

CPU 2

1. CISC
   Complex Instruction Set Computer x86
2. RISC
   Reduced Instruction Set Computing ARM RISC-V

a = a + b 4 a b a+b a

1. CISC(x86) 4 CPU
2. RISC “ ” CPU a b a



ARM (RISC Reduced Instruction Set Computing)

1.
2. CPU
3. RISC CPU

5.2.1

a

1. CPU

```
mov r0, #addr_a //   a    CPU   r0
ldr r1, [r0]    //  r0        CPU   r1
```

2. CPU r1
3. CPU

```
add r1, r1, r2 //  CPU   r1=r1+r2
```

4. CPU

```
str r1, [r0]   //  r1   r0
```

mov add ldr str “ ”— mov

ARM

cond	op1	op	
not 1111	00x	-	MOV
not 1111	010	-	/ LDR/STR
not 1111	011	0	/ LDR/STR
not 1111	011	1	(Media instructions)

cond	op1	op	
not 1111	10x	-	B BL; LDM/STM POP/PUSH
not 1111	11x	-	
1111	-	-	BL

: ARM® and Thumb®-2 Instruction Set Quick Reference Card.pdf (ARM)

: 00_UserManual\ \Arm \ ARM® and Thumb®-2 Instruction Set Quick Reference

5.2.2

```
label
instruction @ comment
```

label label /

instruction

@ comment @ comment

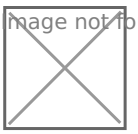
```
add:
mov r0, #0 @ R0 0
```

addmov r0, #0@ R0 0

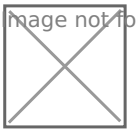
mov bl/b add/sub ldm/stm push/pop

5.2.2.1 mov

```
mov r1, #10 @ 10 r1 r1=10
```



1.
addrA e3a0100a mov r1, #10
2.
MOV
3.
CPU R1 10
e3a0100a MOV



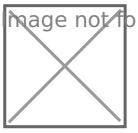
[31:28] 0xe [15 12] R1 0x1 [12:0] 10 0x00a

5.2.2.2 bl

```
1 bl test_tag
2 mov r1, #10
3
4 test_tag:
5 mov r3, #0
6 mov pc, lr
```

1 test_tag mov r3, #0 mov r1, #10 LR

6 mov r1, #10 mov r1, #10



1. CPU addrA eb000000 bl test_tag PC test_tag addrA+8 tes
2. CPU addrA+8 e3a03000 mov r3, #0 CPU R3 0
3. CPU addrA+12 e1a0f00e mov pc, lr PC addrA+4
4. CPU addrA+4 e3a0100a mov r1, #10 CPU R1 10

eb000000 BL

imm[23:0] PC 4 0 ARM ARM addrA bl test_

5.2.2.3 b

```
1 b test_tag
2 mov r1, #10
3
4 test_tag:
5 mov r3, #0
```

1 test_tag mov r3, #0 mov r1, #10

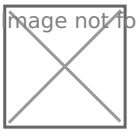
B BL BL B/BL LR BL B

5.2.2.4 add/sub

```
1 mov r1, #10
2 add r2, r1, #4
3 sub r2, r1, #4
```

1 r1 4 r2

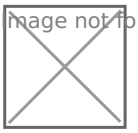
2 r1 4 r2



CPU addrA+4 e2812004 add r2, r1, #4 CPU R2 14

CPU addrA+8 e2412004 sub r2, r1, #4 CPU R2 6

e2812004 ADD



[19: 16] R1 1 [15: 12] R2 2 [11: 0] 4 0x004

e2412004 SUB

[19: 16]　　R1　1 [15: 12]　　R2　2 [11: 0]　　4　0x004

5.2.2.5 ldr/str

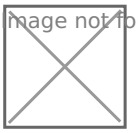
```
1 mov r0, #400H @ 0x400
2 mov r1, #aH   @ 0xa
3 str r1, [r0]
4 ldr r2, [r0]
```

3　　R1　0xa　　R0　　0x400

4　　R0　　0x400　　R2

1. CPU　　addrA　　e3a00b01　mov r0, #400H　　CPU　　R0　0x400
2. CPU　　addrA+4　　e3a0100a　mov r1, #aH　　CPU　　R1　0xa
3. CPU　　addrA+8　　e5801000　str r1, [r0]　　R1 0xa　　R0　　0x400　　0x40
4. CPU　　addrA+12　　e5902000　ldr r2, [r0]　　R0　　0x400　　CPU　　R2　CPU

e5801000 STR



[19: 16]　　R0　0 [15: 12]　　R1　1

e5902000 LDR

[19: 16]　　R0　0 [15: 12]　　R2　2

```
ldr sp,=0x80200000
```

ARM　　0x80200000　　sp　sp=0x80200000

ldr sp,=0x80200000 0x80200000 addrA+4 LDR SP 0x80200(

ldr sp, [pc, #-4] e51fd004 LDR imm12[11: 0] Rn

5.2.2.6 ldm/stm

ldm

stm

```
ldm{cond} Rn{!}, reglist
stm{cond} Rn{!}, reglist
```

cond

IA : 4 STMIA R0,{R1,LR} R1 LR

IB : 4

DA : 4 STMDA R0,{R1,LR} LR R1

DB : 4

FD :

FA :

ED :

EA :

Rn

Rn

reglist , {R1,R2,R6-R9}

```
1 ldr r1,=0x10000000
2
3 ldmib r1!, {r0,r4-r6}
4 stmda r1!, {r0,r4-r6}
```

1　　0x10000000　r1

3　　ib　　4

0X10000004　　R0

0X10000008　　R4

0X1000000C　　R5

0X10000010　　R6

！　　R1　R1=0X10000010

4　　da　　4

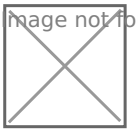
R6　0X10000010

R5　0X1000000C

R4　0X10000008

R0　0X10000004

！　　R1　R1=0X10000000




```
1 ldr sp,=0x80200000
2
3 stmfd sp!, {r0-r2} @
4 ldmfd sp!, {r0-r2} @
```

1　0x80200000　sp

3

R2 0X80200000

R1 0X801FFFFC

R0 0X801FFFF8

4

0X801FFFF8 R0

0X801FFFFC R1

0X80200000 R2

3 4 push pop

```
1 ldr sp,=0x80200000
2
3 push {r0-r2} @
4 pop {r0-r2}  @
```

5.3

5.3.1

0~F 0x H

```
0xA AH
```

0~9 D

```
10 10D
```

0~7　　0　　O

```
012 120
```

0~1　　0b　　B

```
0b1010 1010B
```

5.3.2 C

```
 :int a = 0xA;   // 0x
:  int a = 10;
:  int a = 012;   // 0
:  int a = 0b1010;// 0b
```

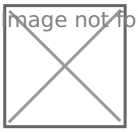
5.3.3

2/16　　8 4 2 1 ——>

0b01101110101

0x375

0xABC1



1010 1011 1100 0001

5.4 /

5.4.1 /

Big-endian

Little-endian

0x12345678 /

addr+3	0x78	0x12
addr+2	0x56	0x34
addr+1	0x34	0x56
addr	0x12	0x78

5.4.2

5.4.2.1

```
1 int a = 0x6; //    0b0110
2 int b = a<<1;
3 int c = a>>1;
```

2 a 0b0110->0b1100 b=0xC

3 a 0b0110->0b0011 b=0x3

5.4.2.2

```
1 int a = 0x6; //    0b0110
2 int b = ~a;
```

2 a 0b0110->0b1001 b=0x9

5.4.2.3

1 1

```
1 int a = 0x6; //    0b0110
2 int b = 0x7; //    0b0111
3
```

```
4 int c = a&b;
```

4 a&b 0b0110 c=0x6

5.4.2.4

1 1

```
1 int a = 0x6; //    0b0110
2 int b = 0x7; //    0b0111
3
4 int c = a| b;
```

4 a|b 0b0111 c=0x7

5.4.2.5

```
1 int a = 0x6;     //    0b0110
2
3 int a |= (1<<3);
```

3 a bit3 1 1<<3 = 0b1000 0b1000|0b0110=0b1110 a=0xe

5.4.2.6

```
1 int a = 0x6;     //    0b0110
2
3 int a &= ~(1<<2);
```

3 a bit2 ~(1<<2) = 0b1011 0b1011&0b0110=0b0010 a=0x2

5.5 C

C ARM ATPCS ATPCS

:

- ATPCS.pdf
- The ARM-THUMB Procedure Call Standard
- : 00_UserManual\ \Arm \ ATPCS(ATM-Thumb).pdf
- :

5.5.1 ATPCS

ATPCS ARM-THUMB procedure call standard ARM-Thumb ARM THUMB

R0~R15 ATPCS

- R0~R3 R0~R3
- R4~R11
- R12 scratch
- R13 SP R13 SP
- R14 LR R14
- R15 PC

5.5.2 C

- 4 R0~R3
- 4

5.5.3 C

- 32 R0
- 64 R0 R1 .
- f0 d0 s0
- f0-fN d0~dN
-

5.5.4 C

/

- /

CPU R0~R3 LR

, CPU

push

pop

-

4 4 4

5.6 C

volatile CCM_CCGR1 32 unsigned int *

```
volatile unsigned int *CCM_CCGR1 = (volatile unsigned int *)(0x20C406C);
```

```
val = *CCM_CCGR1;       //
*CCM_CCGR1 |= (3<<30);  //      CCM_CCGR1   [31 30]  1
```

5.7 start.S

```
2 .text
3 .global _start
4 _start:
```

- 2 .text
- 3 .global _start
- 4 _ start _ start ENTRY C main() _ start

```
@       _start
4 _start:
5
6  //
7  ldr sp,=0x80200000
8
9  bl clean_bss
10
11 bl main
12
13 halt:
```

```
14 b halt
```

- 7 0x80200000 sp C /
- 9 clean_bss clean_bss bl main lr
- 11 C main() b halt lr
- 13 halt
- 14 halt b halt

```
@       clean_bss
16 clean_bss:
17 /*   BSS  */
18 ldr r1, =__bss_start
19 ldr r2, =__bss_end
20 mov r3, #0
21 clean:       @             R1 R2
22 str r3, [r1]
23 add r1, r1, #4
24 cmp r1, r2
25 bne clean
26
27 mov pc, lr @
```

- 16 clean_bss BSS BSS 0
- 18 bss r1
- 19 bss r2
- 20 0 r3 r3=0
- 21 clean
- 22 r3 r1
- 23 r1 4 r1 r1 = r1+4
- 24 r1 r2
- 25 r1 r2 clean
- 26 r1 r2 bl main

5.8 led.dis

led.dis imx6ull led.bin DDR

imx6ull bootRom

Image not found or type unknown

bootRom

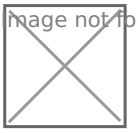
1. bootRom EMMC TF 4K RAM
2. bootRom DCD DDR
3. bootRom IVT EMMC TF led.bin DDR 0x80100000
4. DDR 0x80100000

led.bin CPU 0x80100000 32 /4 led.bin led.l

/ e59fd028 ldr sp,=0x80200000

00000000	28
00000001	d0
00000002	9f
00000003	e5

imx6ARM bin objdump led.dis :



led.dis 1.7 start.S

```
    1)  CPU           0x80100000   e59fd028        ldr sp, [pc, #40]    Start.S   ldr sp,=0x802
```

```
80100000:  e59fd028 ldr sp, [pc, #40] ; 80100030 <clean+0x14>
```

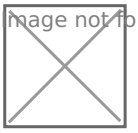
2. 0x80100004 eb000001 bl 80100010 Start.S bl clean_bss

```
80100004:  eb000001 bl 80100010 <clean_bss>
....
80100010 <clean_bss>:
80100010:  e59f101c ldr r1, [pc, #28] ; 80100034 <clean+0x18>
80100014:  e59f201c ldr r2, [pc, #28] ; 80100038 <clean+0x1c>
```

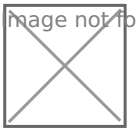
3. 0x80100010 e59f101c ldr r1, [pc, #28] Start.S ldr r1, =__bss_start
4. clean_bss CPU 4 mov pc, lr 0x80100008 fa000057

CPU C main()

C main()



1. main() R7 LR / main() R7



2. led_init() BL bl 8010003c

3. led_ctl(1) R0 movs r0, #1 BL bl 801000f8

4. delay(1000000) R0 BL

5. led_ctl(0) R0 BL

6. delay(1000000) R0 BL

7. while(1) B b.n 80100174 0x80100174 led

main() led_init() led_ctl() delay()