ARM Notes From Youtube

Tags	ARM
Created time	@November 22, 2022 4:01 PM

```
//ADDRESS MODE
.section .data
.align 2
.global _start
_start:
  LDR R0,=list
  LDR R1,[R0]
  LDR R2,[R0,#4]!
  LDR R3,[R0], #4

  MOV R7,#1
  SWI 0



.section .data
_list:
  .word 4, 5, -9, 1, 0, 2, -3

//list = [4, 5, -9, 1, 0, 2, -3]
//list[R0]

//list[R0]
```

```
// CPSR ARITHMETIC
.global _start
_start:
  MOV R0, #5
  MOV R1, #7
  ADD R2, R0, R1 //R2 = R0 + R1

  SUB R3, R2, R1 // R3 = R2 - R1
  SUBS R4, R1, R2 // R4 = R1 - R2 //trigger cpsr,

  MOV R6, #0xffffffff
  MOV R8, #0x4
  SUB R5, R6, R8
  ADDS R9, R6, R8 //TRIGGER CPSR, may overflow which has carry
  ADC R10, R0, R1 //ADD CARRY
  MOV R7,#1
  SWI 0
```

```
  //SUB
  //MUL
```

```
//LOGICAL OPERATION
.global _start
_start:
  MOV R0,#0XFF
  MOV R1,#22
  AND R2,R1,R0
  ORR R3,R1,R0
  EOR R4,R1,R0

  MVN R5,R0

  MOV R7,#1
  SWI 0
```

```
//SHIFT AND ROTATIONS
.global _start
_start:
  //LSL MULTIPLY BY 2
  MOV R0,#10
  MOV R1,R0
  LSL R1,#1
  MOV R2,R1,LSL #1
  ROR R0,#1
  ROR R0,#1
  //LSR DIVIDE BY 2

  //ROR 00000101 => 10000010, the last one shifts to the front

  MOV R7,#1
  SWI 0
```

```
//conditions and branches
.global _start
_start:
  MOV R0,#1
  MOV R1,#2

  CMP R0,R1 //R0-R1
//  CMP R1,R0 //R1-R0

  BGT greater
  BAL default

//  BEQ
//  BNE
//  BLT
//  BLE
```

```
//  BGE

greater:
  MOV R2,#1

default:
  MOV R2,#2
```

```
// LOOPS WITH BRANCHES
.global _start
.equ endlist, 0xaaaaaaaa

_start:
  LDR R0,=list
  LDR R1,[R0]
  LDR R3,=endlist
  ADD R2,R2,R1
  MOV R2,#0

loop:
  LDR R1,[R0,#4]!
  CMP R1,R3
  BEQ exit
  ADD R2,R2,R1
  BAL loop

exit:

.data
list:
  .word 1,2,3,4,5,6,7,8,9,10
```

```
//CONDITIONAL INSTRUCTIONS EXECUTION
  MOV R0,#2
  MOV R1,#3
  CMP R1,R0

  MOVGT R2,#10
  //ADDLT LESS THAN
  //MOVGE GREATER AND EQUAL
```

```
link register and return(function)
.global _start
.equ endlist, 0xaaaaaaaa

_start:

  MOV R0,#2
  MOV R1,#3
  BL add2
```

```
  MOV R3,#4

add2:
  ADD R2,R0,R1
  mov pc, lr
```

```
// push and pop using stack memory

  MOV R0,#2
  MOV R1,#3
  PUSH {R0,R1}
  BL add2
  POP {R0,R1}
  MOV R3,#4

add2:
  MOV R0,#5
  MOV R1,#7
  ADD R2,R0,R1
  BX lr
```

```
.equ SWITCH, 0xff200040
.equ LED, 0xff200000
.equ SEVEN, 0xff200020
.global _start
_start:
  LDR R0,=SWITCH
  LDR R1,[R0]

  LDR R0,=LED
  STR R1,[R0]

  LDR R0,=SEVEN
  STR R1,[R0]
```