



**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**



**BÀI GIẢNG MÔN**

# **KỸ THUẬT VI XỬ LÝ**

**Giảng viên:**

**Điện thoại/E-mail:**

**Học kỳ:**

**TS. Nguyễn Trung Hiếu**

**0916566268 / [hieunt@ptit.edu.vn](mailto:hieunt@ptit.edu.vn)**

**Kỳ 1/2020**



# CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

## NỘI DUNG

1. Giới thiệu về lập trình Assembly
2. Tập lệnh của ARM
  1. Lệnh xử lý dữ liệu (data processing)
  2. Dịch chuyển dữ liệu (data movement)
  3. Điều khiển chương trình (flow control)
  4. Ngắt

# CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

## ■ Cấu trúc chung của một chương trình hợp ngữ:

```
AREA Example, CODE, READONLY      ; name this block of code
ENTRY                             ; mark first instruction
                                   ; to execute

start
MOV    r0, #15                    ; Set up parameters
MOV    r1, #20
BL     firstfunc                  ; Call subroutine
SWI     0x11                      ; terminate

firstfunc                          ; Subroutine firstfunc
ADD     r0, r0, r1                ; r0 = r0 + r1
MOV     pc, lr                    ; Return from subroutine
                                   ; with result in r0
END                                     ; mark end of file
```

The diagram illustrates the structure of an ARM assembly program. It shows a sequence of instructions with labels, opcodes, operands, and comments. The labels 'start' and 'firstfunc' are on the left, with arrows pointing to the first and fourth instructions respectively. The instructions are: 'MOV r0, #15', 'MOV r1, #20', 'BL firstfunc', and 'SWI 0x11'. The first instruction is labeled 'start'. The fourth instruction is labeled 'firstfunc'. The instruction 'BL firstfunc' is underlined. The instruction 'MOV pc, lr' is underlined. The instruction 'END' is underlined. The instruction 'MOV pc, lr' is labeled 'operands'. The instruction 'END' is labeled 'opcode'. The instruction 'END' is labeled 'label'. The instruction 'END' is labeled 'comment'.

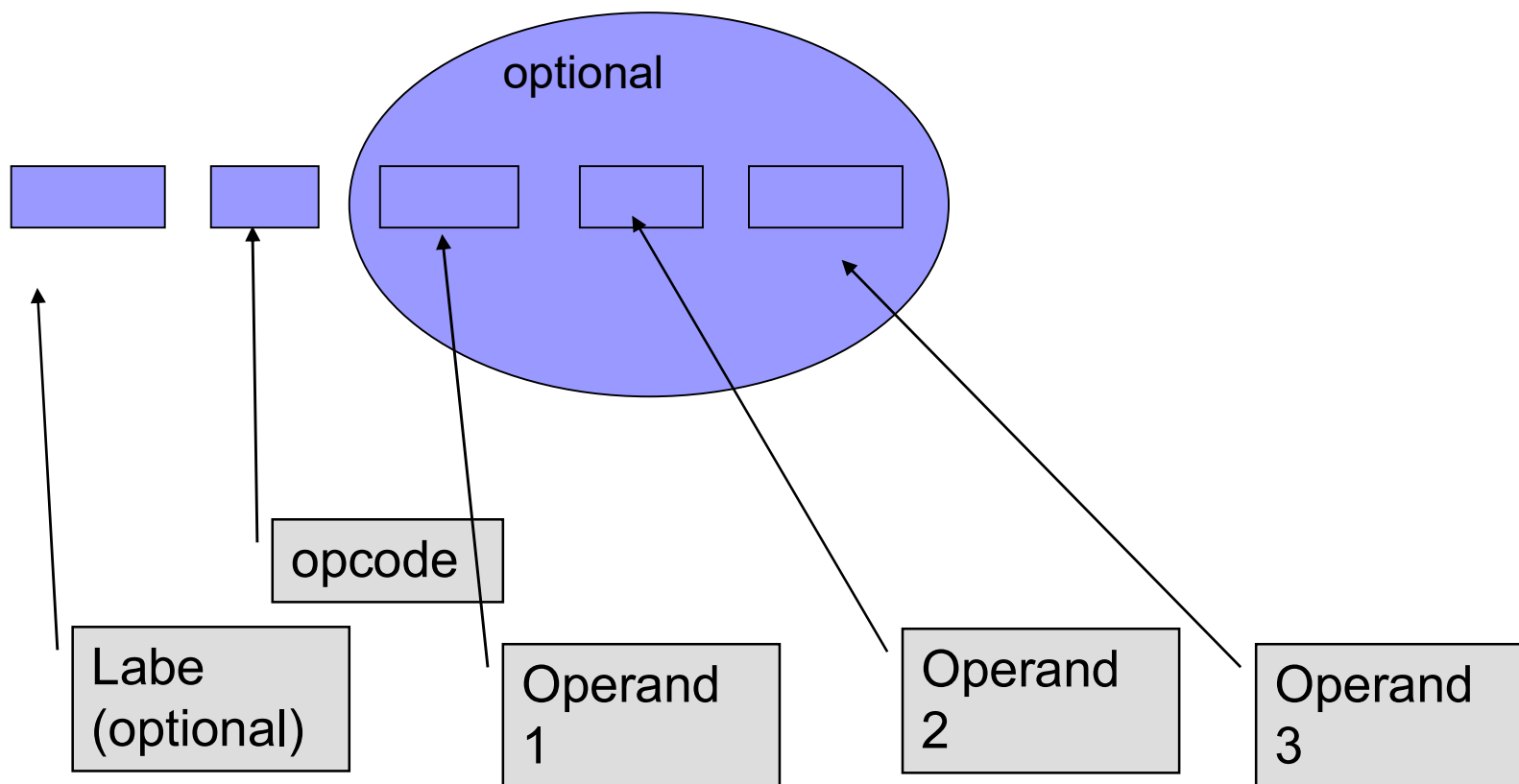
Diagram labels and arrows:

- label**: points to the 'start' label.
- opcode**: points to the 'END' instruction.
- operands**: points to the 'pc, lr' operands in the 'MOV pc, lr' instruction.
- comment**: points to the ';' comments in the instructions.

## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

### ■ Cấu trúc chung của một lệnh ARM

*label <dấu cách> lệnh <dấu cách> ; chú giải*





# CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

## NỘI DUNG

1. Giới thiệu về lập trình Assembly
2. Tập lệnh của ARM
  1. Lệnh xử lý dữ liệu (data processing)
  2. Di chuyển dữ liệu (data transfer)
  3. Điều khiển chương trình (flow control)



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

- Đặc điểm của tập lệnh ARM:
  - Kiến trúc Load – Store
  - Lệnh 3 địa chỉ
  - Tất cả đều là cách lệnh có điều kiện
  - Có khả năng load/store nhiều thanh ghi đồng thời
  - Khối dịch và khối ALU có thể hoạt động song song, do đó phép tính dịch và các phép tính tính toán có thể được thực hiện đồng thời.



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

### ■ Lệnh xử lý dữ liệu:

- Lệnh xử lý dữ liệu bao gồm lệnh di chuyển dữ liệu giữa các thanh ghi (move), lệnh số học (arithmetic), lệnh logic (logical), lệnh so sánh (comparison) và lệnh nhân (multiply).
- Hầu hết các lệnh xử lý dữ liệu có thể dùng bộ dịch (barrel shifter) để xử lý một trong những toán hạng của lệnh



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

- Lệnh xử lý dữ liệu
  - Arithmetic:      **ADD    ADC    SUB    SBC    RSB    RSC**
  - Logical:          **AND    ORR    EOR    BIC**
  - Comparisons:    **CMP    CMN    TST    TEQ**
  - Data movement: **MOV    MVN**
- Lưu ý: các lệnh chỉ thực hiện trên thanh ghi, KHÔNG thực hiện trên bộ nhớ.
- Cú pháp:  
    **<Operation>{<cond>}{S} Rd, Rn, Operand2**
  - Lệnh so sánh tác động đến cờ và thanh ghi Rd không bị tác động
  - Lệnh di chuyển dữ liệu không tác động đến thanh ghi Rn
- Toán hạng thứ 2 được đưa đến ALU thông qua bộ dịch chuyển





## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

- Di chuyển dữ liệu giữa các thanh ghi

*MOV<cond><S> Rd, Rn, <operands>*

MOVCS R0, R1 ; Nếu cờ nhớ C = 1 thì R0 := R1

MOVS R0, #0 ; R0 = 0  
; Z = 1, N = 0  
; C, V không bị tác động

## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

- Di chuyển dữ liệu giữa các thanh ghi:
  - Lưu ý: Hầu hết các lệnh trong ARM có trường điều kiện. Khi thỏa mãn điều kiện đó thì lệnh mới được thực hiện.

*MOVCS R0, R1 ; R0 = R1 chỉ khi C = 1.*

*MOVCC R0, R1 ; R0 = R1 chỉ khi C = 0.*

| Mnemonic | Condition                      | Mnemonic | Condition                     |
|----------|--------------------------------|----------|-------------------------------|
| CS       | <i>Carry Set</i>               | CC       | <i>Carry Clear</i>            |
| EQ       | <i>Equal (Zero Set)</i>        | NE       | <i>Not Equal (Zero Clear)</i> |
| VS       | <i>Overflow Set</i>            | VC       | <i>Overflow Clear</i>         |
| GT       | <i>Greater Than</i>            | LT       | <i>Less Than</i>              |
| GE       | <i>Greater Than or Equal</i>   | LE       | <i>Less Than or Equal</i>     |
| PL       | <i>Plus (Positive)</i>         | MI       | <i>Minus (Negative)</i>       |
| HI       | <i>Higher Than</i>             | LO       | <i>Lower Than (aka CC)</i>    |
| HS       | <i>Higher or Same (aka CS)</i> | LS       | <i>Lower or Same</i>          |

## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

### ■ Di chuyển dữ liệu giữa các thanh ghi:

- Gồm có hai lệnh:

- *MOV R0, R2* ;  $R0 = R2$

- *MVN R0, R2* ; *move negative,  $R0 = -R2$*

- Ví dụ:

*Trước:*  $r5 = 5$

$r7 = 8$

*MOV r7, r5*

*Sau:*  $r5 = 5$

$r7 = 5$



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

- Di chuyển dữ liệu giữa các thanh ghi:

`MOV R1, #0x12`

; Lưu số 8 bit vào phần thấp của R1

`MOV R1, #0x1234`

; Lưu số 16 bit vào phần thấp của R1

`MOVT R1, #0x1234`

; Lưu số 16 bit vào phần cao của R1

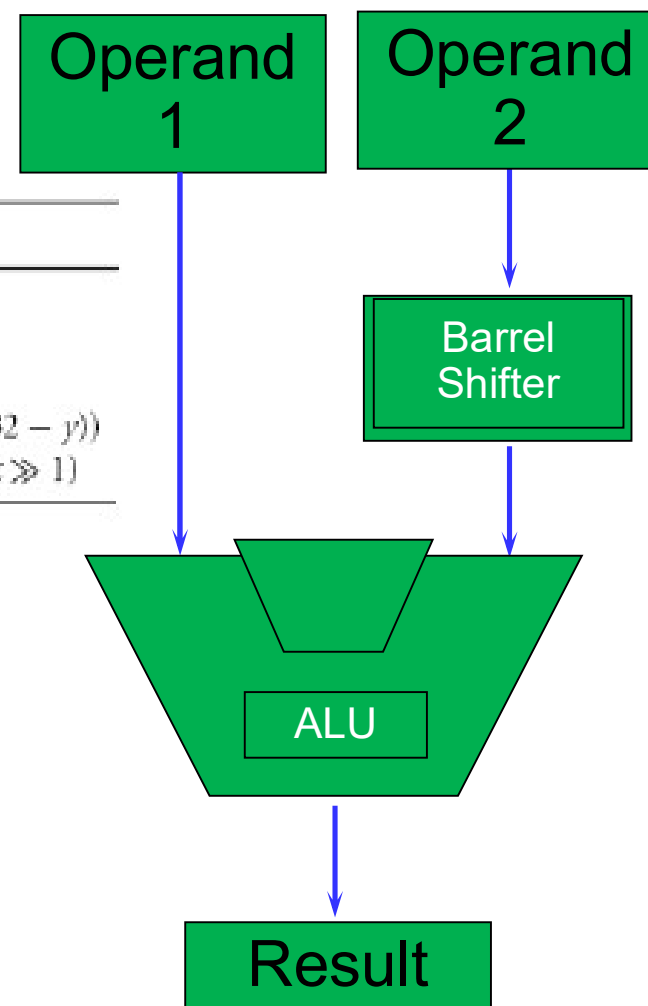
`MOV32 R1, #0x12345678`

; Lưu số 32 bit vào R1

# CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

## ■ Barrel shifter:

| Mnemonic | Description            | Shift            | Result  |
|----------|------------------------|------------------|---|
| LSL      | logical shift left     | $x\text{LSL } y$ | $x \ll y$   |
| LSR      | logical shift right    | $x\text{LSR } y$ | $(\text{unsigned})x \gg y$                            |
| ASR      | arithmetic right shift | $x\text{ASR } y$ | $(\text{signed})x \gg y$                              |
| ROR      | rotate right           | $x\text{ROR } y$ | $((\text{unsigned})x \gg y)   (x \ll (32 - y))$       |
| RRX      | rotate right extended  | $x\text{RRX}$    | $(c\text{ flag} \ll 31)   ((\text{unsigned})x \gg 1)$ |



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

### ■ Dịch trái:



- MOV R0, R2, **LSL #2** ; R0 = R2<<2  
; R2 không đổi

Ví dụ: **0...0 0011 0000**

Trước: R2 = 0x00000030

Sau: R0 = 0x000000C0

R2 = 0x00000030

## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

### ■ Dịch phải:



- MOV R0, R2, **LSR #2** ; R0 = R2<<2  
; R2 không đổi

Ví dụ: **0...0 0011 0000**

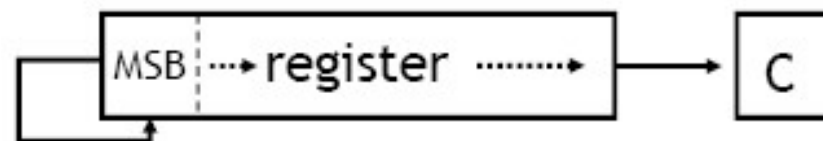
Trước: R2 = 0x00000030

Sau: R0 = 0x0000000C

R2 = 0x00000030

## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

### ■ Dịch phải số học:



- MOV R0, R2, **ASR #2** ; R0 = R2>>2  
; R2 không đổi

Ví dụ: **1010 0...0 0011 0000**

Trước: R2 = 0xA0000030

Sau: R0 = 0xE800000C

R2 = 0xA0000030



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

### ■ Quay phải mở rộng:



- MOV R0, R2, **RRX** ; R0 = R2 sau khi quay  
; R2 không đổi

Ví dụ: **0...0 0011 0001**

Trước: R2 = 0x00000031 ; C = 1

Sau: R0 = 0x80000018 ; C = 1

R2 = 0x00000031

## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

### ■ Các lệnh số học: Cộng và trừ

*instruction* <cond> <S> *Rd*, *Rn*, *N*

|     |  |                                      |
|-----|--|--------------------------------------|
| ADC | add two 32-bit values and carry                  | $Rd = Rn + N + \text{carry}$         |
| ADD | add two 32-bit values                            | $Rd = Rn + N$                        |
| RSB | reverse subtract of two 32-bit values            | $Rd = N - Rn$                        |
| RSC | reverse subtract with carry of two 32-bit values | $Rd = N - Rn - !(\text{carry flag})$ |
| SBC | subtract with carry of two 32-bit values         | $Rd = Rn - N - !(\text{carry flag})$ |
| SUB | subtract two 32-bit values                       | $Rd = Rn - N$                        |



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

- Các chế độ địa chỉ:
  - Chế độ địa chỉ thanh ghi:
    - `ADD R0, R1, R2` ; Các toán hạng là các thanh ghi
  - Chế độ địa chỉ tức thời:
    - `ADD R3, R3, #1` ;  $R3 = R3 + 1$
    - `ADD R8, R7, #0xff` ;  $R8 = R7[7:0]$
- Dấu # được sử dụng để biểu diễn toán hạng tức thời
- Toán hạng có ký hiệu `0x` đứng trước: biểu diễn số hexa



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

### ■ Các lệnh số học: Cộng và trừ

Ví dụ 1:

R0 = 0x00000000

R1 = 0x00000002

R2 = 0x00000001

SUB R0, R1, R2

Ví dụ 2:

R0 = 0x00000000

R1 = 0x00000077

RSB R0, R1, #0



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

### ■ Các lệnh số học: Cộng và trừ

Ví dụ 3:

R1 = 0x00000001

SUBS R1, R1, #1

CPRS = ?

Ví dụ 4:

R0 = 0x00000000

R1 = 0x00000005

ADD R0, R1, R1, LSL #1



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

- Các lệnh số học: Cộng và trừ

Ví dụ 5: Thực hiện phép cộng 3 số sau:

Số thứ nhất = 2ABC3458

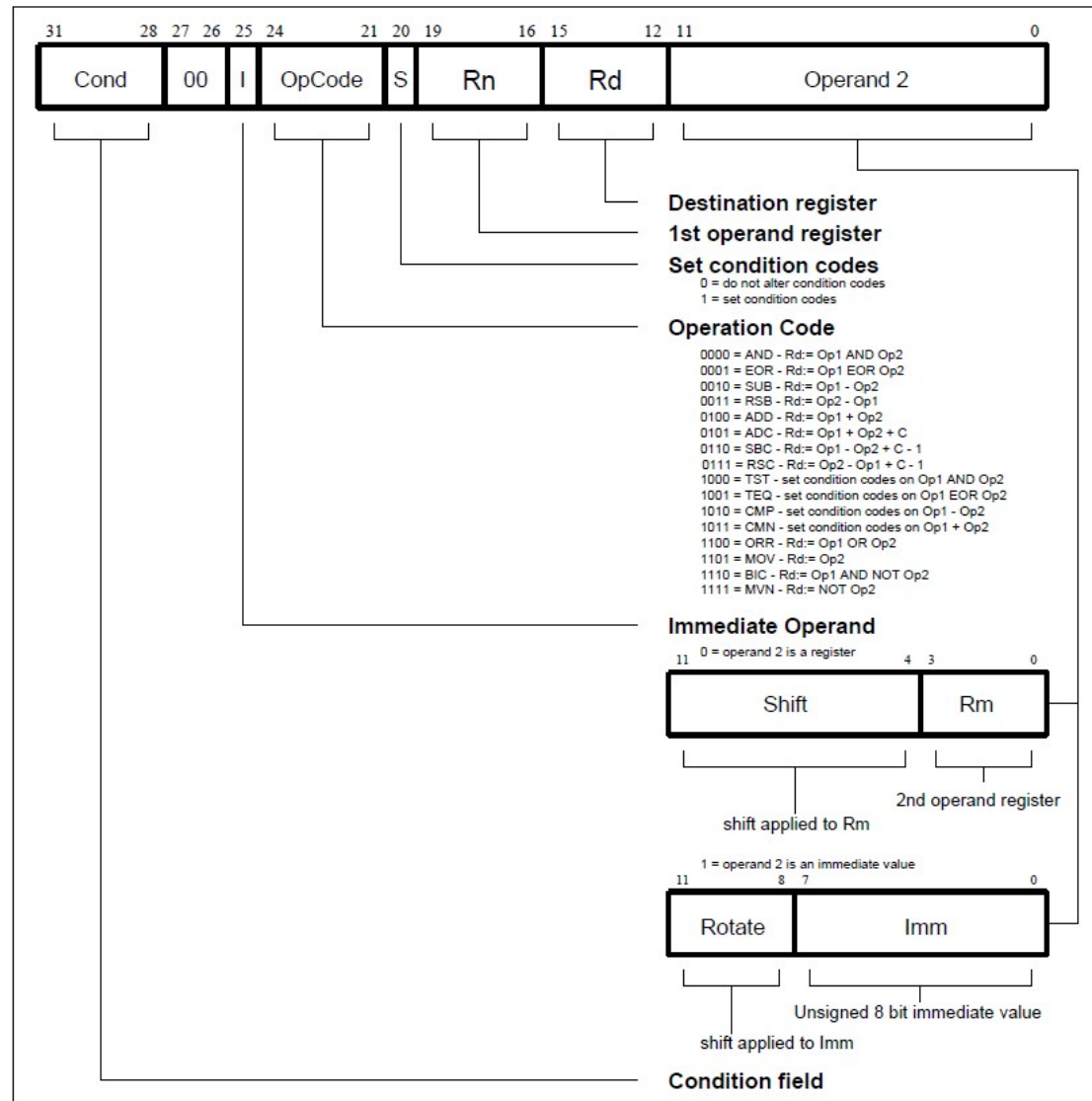
Số thứ hai = E8934D47

Số thứ ba = 134DC378

Kết quả lưu vào thanh ghi R4

# CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

- Cấu trúc lệnh được mã hóa:



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

### ■ Cấu trúc lệnh: Condition

| Code | Suffix | Flags   | Meaning                 |
|------|--------|---------|-------------------------|
| 0000 | EQ     | Z set   | equal                   |
| 0001 | NE     | Z clear | not equal               |
| 0010 | CS     | C set   | unsigned higher or same |
| 0011 | CC     | C clear | unsigned lower          |
| 0100 | MI     | N set   | negative                |
| 0101 | PL     | N clear | positive or zero        |
| 0110 | VS     | V set   | overflow                |
| 0111 | VC     | V clear | no overflow             |



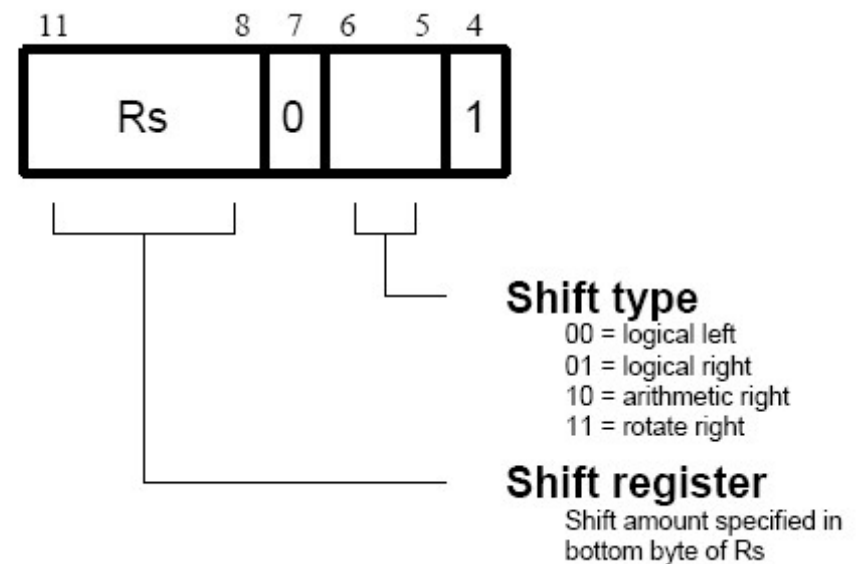
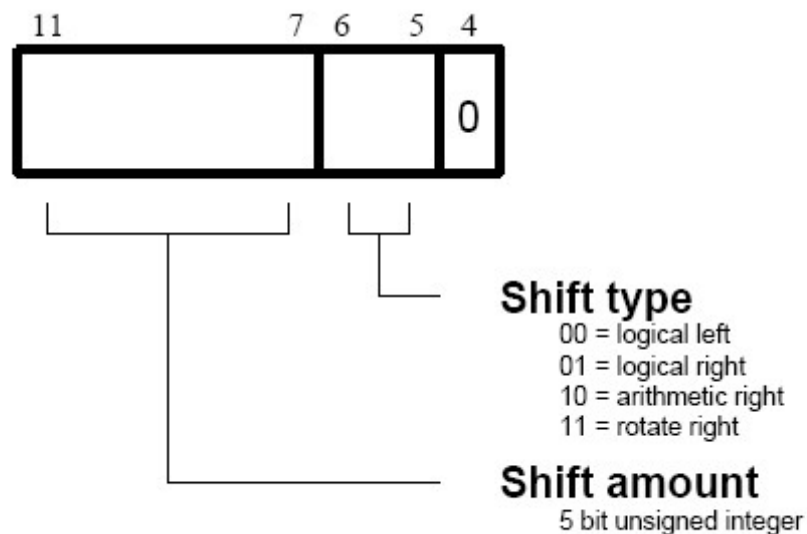
# CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

## ■ Cấu trúc lệnh: Opcode

| Assembler Mnemonic | OpCode | Action                            |
|--------------------|--------|-----------------------------------|
| AND                | 0000   | operand1 AND operand2             |
| EOR                | 0001   | operand1 EOR operand2             |
| SUB                | 0010   | operand1 - operand2               |
| RSB                | 0011   | operand2 - operand1               |
| ADD                | 0100   | operand1 + operand2               |
| ADC                | 0101   | operand1 + operand2 + carry       |
| SBC                | 0110   | operand1 - operand2 + carry - 1   |
| RSC                | 0111   | operand2 - operand1 + carry - 1   |
| TST                | 1000   | as AND, but result is not written |
| TEQ                | 1001   | as EOR, but result is not written |

# CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

## ■ Cấu trúc lệnh: Operand2





## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

### ■ Cấu trúc lệnh:

Lệnh:

MOV R4,#0x00

MOV R5,#0x00

MOV R7,#0x02

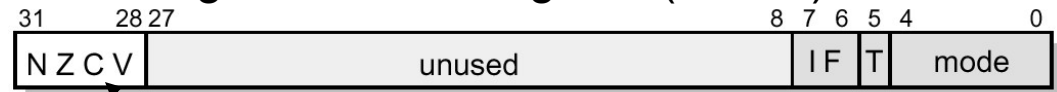
ADDEQ R4,R5,R7,LSR #2

# Exercise 2

Fill in the shaded areas.

Program counter PC =R15, #value = intermediate constant value

Current Program Status Register (CPSR)



| Address (H) |  | Comments                        | After instruction is run |   |           |           |           |
|-------------|--|---------------------------------|--------------------------|---|-----------|-----------|-----------|
|             |  |                                 | PC (Hex)                 | C | R0(Hex)   | R1(Hex)   | R2 (Hex)  |
| PC          |  |                                 |                          |   |           |           |           |
|             | All registers R0-R2 are rest to 0 here |                                 |                          | 0 | 0         | 0         |           |
| 0000 1000   | Mov r1,#15                             | ;r1=15                          | 0000 1004                | 0 | 0000 0000 | 0000 000f | ffff ffff |
|             | Mov r2,#0xffffffff                     | ;r2=#0xffffffff<br>;i.e. r2= -1 |                          |   |           |           |           |
|             | ADDs r0,r1,r2                          | ;r0=r1+r2                       |                          |   |           |           |           |
|             | ADCs r0,r1,r2                          | ;r0=r1+r2+C                     |                          |   |           |           |           |
|             | SUBs r0,r1,r2                          | ;r0=r1-r2                       |                          |   |           |           |           |
|             | SBCs r0,r1,r2                          | ;r0=r1-r2+C-1                   |                          |   |           |           |           |



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

### ■ Lệnh logic

instruction<cond><S> Rd, Rn, N

|     |   |                     |
|-----|---|---------------------|
| AND | logical bitwise AND of two 32-bit values  | $Rd = Rn \& N$      |
| ORR | logical bitwise OR of two 32-bit values   | $Rd = Rn   N$       |
| EOR | logical exclusive OR of two 32-bit values | $Rd = Rn \wedge N$  |
| BIC | logical bit clear (AND NOT)               | $Rd = Rn \& \sim N$ |



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

### ■ Lệnh logic

- AND     R0, R1, R2     ; R0 = R1 and R2
- ORR     R0, R1, R2     ; R0 = R1 or R2
- EOR     R0, R1, R2     ; R0 = R1 xor R2
- BIC     R0, R1, R2     ; R0 = R1 and ( $\sim$ R2)

Lệnh BIC là lệnh xóa bit: Các bit trong R1 sẽ bị xóa bởi các bit được đánh dấu trong R2

R1 = 0x11111111    R2 = 0x01100101

BIC R0, R1, R2

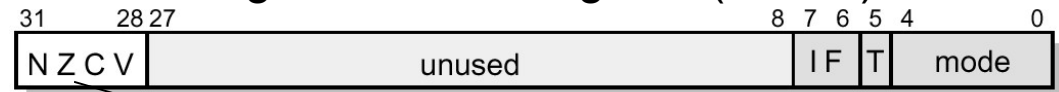
R0 = 0x10011010

# Exercise 3

Fill in the shaded areas.

Program counter PC =R15, #value = intermediate constant value

Current Program Status Register (CPSR)



| Address (H) |                  | Comments                    | After instruction is run |            |            |    |
|-------------|------------------|-----------------------------|--------------------------|------------|------------|----|
| PC          |                  |                             | R0(Hex)                  | R1(Hex)    | R2(Hex)    | NZ |
|             | At the beginning |                             | 0000 0000H               | 0000 0055H | 0000 0061H | 00 |
| 0000 7000   | ANDs r0,r1,r2    | ;r0=r1 and r2 (bit by bit ) |                          |            |            |    |
|             | ORRs r0,r1,r2    | ;r0=r1 or r2                |                          |            |            |    |
|             | EORs r0,r1,r2    | ;r0=r1 xor r2               |                          |            |            |    |
|             | BICs r0,r1,r2    | ;r0=r1 and (not r2)         |                          |            |            |    |

R1=55H=0101 0101 B

R2=61H=0110 0001 B

9EH=1001 1110 B

## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

### ■ Lệnh so sánh

- Các lệnh so sánh không tạo ra kết quả nhưng nó tác động đến các bit cờ (N, Z, C, V) trong thanh ghi CPSR.
- Cú pháp: instruction<cond> Rn, N

|     |  |  |
|-----|--|--|
| CMN | compare negated                        | flags set as a result of $Rn + N$      |
| CMP | compare                                | flags set as a result of $Rn - N$      |
| TEQ | test for equality of two 32-bit values | flags set as a result of $Rn \wedge N$ |
| TST | test bits of a 32-bit value            | flags set as a result of $Rn \& N$     |





## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

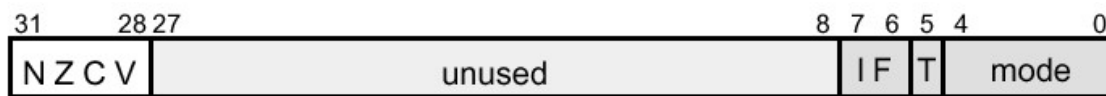
### ■ Lệnh so sánh

- CMP R1, R2 ; Thiết lập cờ dựa trên kết quả  $R1 - R2$
- CMN R1, R2 ; Thiết lập cờ dựa trên kết quả  $R1 + R2$
- TST R1, R2 ; bit test: Thiết lập cờ dựa trên kq R1 and R2
- TEQ R1, R2 ; test equal: Thiết lập cờ dựa trên kq  $R1 \text{ xor } R2$

# CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

## ■ Lệnh so sánh

**CMP r1, r2 ; set cc on r1 - r2 (compare)**



- Same as SUB (subtract) except result of subtraction is not stored.
- Only the condition code bits (cc) {N,Z,C,V} in CPSR are changed

- N = 1 if MSB of (r1 - r2) is '1' (MSB of result is sign bit, 1 = negative)
- Z=1 when the result is zero
- C=1 when the result of an addition is greater than or equal to  $2^{32}$ , if the result of a subtraction is positive.
- V=1 (when result of add, subtract, or compare is  $\geq 2^{31}$ , or  $< -2^{31}$ ). I.e.
  - if two -ve numbers are added, the result is +ve (underflow).
  - if two +ve numbers are added, the result is -ve (overflow).(0x7FFFFFFF+1=0x80000000)



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

### ■ Lệnh so sánh

Ví dụ:

R1 = a, R2 = b

if (R1<R2)

R1=a

else

R1=b

# CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

## Exercise 6

Fill in the shaded areas.

TST updates the N and Z flags according to the result, It does not affect the C or V flags.

| Address (H) |   | Comments  | After instruction is run |          |          |
|-------------|---|---|--------------------------|----------|----------|
| PC          |   |   | NZCV (binary)            | R1 (Hex) | R2 (Hex) |
|             | All registers R0-R2=0 and NZCV=0000, here |   |                          |          |          |
| 0000 1000   | Mov r1,#15                                | ;r1=15 decimal  |                          |          |          |
|             | Mov r2,#0x240                             | ;r2=0xF0 (0xf is 240 in decimal)                        |                          |          |          |
|             | TST r1,r2                                 | ; set cc on r1 AND r2 (logical AND operation test bits) |                          |          |          |
|             | TEQ r1,r2                                 | ; set cc on r1 xor r2 (test equivalent)                 |                          |          |          |

Convert hex to decimal :<http://easycalculation.com/hex-converter.php>

0000 1111

0000 1111

## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

### ■ Lệnh nhân

MLA<cond><S> Rd, Rm, Rs, Rn

MUL<cond><S> Rd, Rm, Rs

|     |                         |                       |
|-----|-------------------------|-----------------------|
| MLA | multiply and accumulate | $Rd = (Rm * Rs) + Rn$ |
| MUL | multiply                | $Rd = Rm * Rs$        |

- Tất cả các toán hạng phải là thanh ghi
- Thanh ghi Rm, Rs phải là hai thanh ghi khác nhau

## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

### ■ Lệnh nhân 64 bit

instruction<cond><S> RdLo, RdHi, Rm, Rs

|       |                                   |  |
|-------|-----------------------------------|--|
| SMLAL | signed multiply accumulate long   | $[RdHi, RdLo] = [RdHi, RdLo] + (Rm * Rs)$    |
| SMULL | signed multiply long              | $[RdHi, RdLo] = Rm * Rs$                     |
| UMLAL | unsigned multiply accumulate long | $[RdHi, RdLo] = [RdHi, RdLo] \mid (Rm * Rs)$ |
| UMULL | unsigned multiply long            | $[RdHi, RdLo] = Rm * Rs$                     |



# CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

## NỘI DUNG

1. Giới thiệu về lập trình Assembly
2. Tập lệnh của ARM
  1. Lệnh xử lý dữ liệu (data processing)
  2. **Điều khiển chương trình (flow control)**
  3. Di chuyển dữ liệu (data transfer)
  4. Ngắt



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

### ■ Lệnh rẽ nhánh

- Lệnh rẽ nhánh không điều kiện

**B** label

....

label: ....

- Lệnh rẽ nhánh có điều kiện

MOV R0, #0

loop: ADD R0, R0, #1

CMP R0, #10

**BNE** loop



# CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

## ■ Lệnh rẽ nhánh

| Branch     | Interpretation              | Normal uses  |
|------------|-----------------------------|--|
| B BAL      | Unconditional<br>Always     | Always take this branch<br>Always take this branch                             |
| BEQ        | Equal                       | Comparison equal or zero result  |
| BNE        | Not equal                   | Comparison not equal or non-zero result  |
| BPL        | Plus                        | Result positive or zero  |
| BMI        | Minus                       | Result minus or negative   |
| BCC<br>BLO | Carry clear<br>Lower        | Arithmetic operation did not give carry-out<br>Unsigned comparison gave lower  |
| BCS<br>BHS | Carry set Higher<br>or same | Arithmetic operation gave carry-out<br>Unsigned comparison gave higher or same |
| BVC        | Overflow clear              | Signed integer operation; no overflow occurred                                 |
| BVS        | Overflow set                | Signed integer operation; overflow occurred                                    |
| BGT        | Greater than                | Signed integer comparison gave greater than                                    |
| BGE        | Greater or equal            | Signed integer comparison gave greater or equal                                |
| BLT        | Less than                   | Signed integer comparison gave less than                                       |
| BLE        | Less or equal               | Signed integer comparison gave less than or equal                              |
| BHI        | Higher                      | Unsigned comparison gave higher  |
| BLS        | Lower or same               | Unsigned comparison gave lower or same   |

## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

### ■ Lệnh rẽ nhánh

```
CMP    R0, #5
BEQ    bypass      @ if (R0!=5) {
ADD    R1, R1, R0 @  R1=R1+R0-R2
SUB    R1, R1, R2 @ }
```

bypass: ...

```
-----
CMP    R0, #5      smaller and faster
ADDNE  R1, R1, R0
SUBNE  R1, R1, R2
```

Rule of thumb: if the conditional sequence is three instructions or less, it is better to use conditional execution than a branch.

## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

### ■ Lệnh rẽ nhánh

```
if ((R0==R1) && (R2==R3)) R4++
```

-----

```
    CMP    R0, R1
    BNE    skip
    CMP    R2, R3
    BNE    skip
    ADD    R4, R4, #1
```

```
skip:    ...
```

-----

```
    CMP    R0, R1
    CMPEQ  R2, R3
    ADDEQ  R4, R4, #1
```



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

### ■ Lệnh gọi hàm

- Lệnh BL copy địa chỉ quay trở về chương trình chính vào thanh ghi R14 (lr)

BL subfunc ; gọi hàm subfunc

CMP R1, #5 ; vị trí quay trở lại

.....

Subfunc: ..... ; hàm con

.....

MOV PC, LR ; quay lại chương trình chính



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

### ■ Lệnh gọi hàm

- B Label: Rẽ nhánh đến Label
- BL Label: Rẽ nhánh đến Label và lệnh tiếp theo trong LR
- BX R0: Rẽ nhánh đến địa chỉ lưu trong R0
- BLX R0: Rẽ nhánh đến địa chỉ lưu trong thanh ghi R0



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

- Ví dụ: Tìm số lớn nhất trong 3 số sau và lưu trong R3:

R0 = 0x342454AB

R1 = 0x4345CD21

R2 = 0x34CD2345

R3 =?



# CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

## NỘI DUNG

1. Giới thiệu về lập trình Assembly
2. Tập lệnh của ARM
  1. Lệnh xử lý dữ liệu (data processing)
  2. Điều khiển chương trình (flow control)
  3. Di chuyển dữ liệu (data transfer)
  4. Ngắt



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

- Lệnh di chuyển dữ liệu
  - Di chuyển dữ liệu giữa các thanh ghi và bộ nhớ
  - Có 3 dạng chính:
    - Load/store một thanh ghi
    - Load/store nhiều thanh ghi
    - Hoán chuyển dữ liệu giữa ô nhớ và thanh ghi





## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

### ■ Load/Store một thanh ghi

- Cú pháp:
  - `<LDR|STR>{cond} {B|SB|H|SH} Rd, [Rn]` : Nạp dữ liệu từ ô nhớ có địa chỉ lưu trong Rn vào thanh ghi Rd
- Độ dài dữ liệu có thể là 1 byte, 1 byte có dấu, nửa từ, nửa từ có dấu, 1 từ

```
LDR    r0, [r1]    ; r0 := mem32[r1]
STR    r0, [r1]    ; mem32[r1] := r0
```

## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

### ■ Load/Store một thanh ghi

|              |   |   |
|--------------|---|---|
| <b>LDR</b>   | Load a word into register                 | $Rd \leftarrow \text{mem32}[\text{address}]$                    |
| <b>STR</b>   | Store a word in register into memory      | $\text{Mem32}[\text{address}] \leftarrow Rd$                    |
| <b>LDRB</b>  | Load a byte into register                 | $Rd \leftarrow \text{mem8}[\text{address}]$                     |
| <b>STRB</b>  | Store a byte in register into memory      | $\text{Mem8}[\text{address}] \leftarrow Rd$                     |
| <b>LDRH</b>  | Load a half-word into register            | $Rd \leftarrow \text{mem16}[\text{address}]$                    |
| <b>STRH</b>  | Store a half-word in register into memory | $\text{Mem16}[\text{address}] \leftarrow Rd$                    |
| <b>LDRSB</b> | Load a signed byte into register          | $Rd \leftarrow \text{signExtend}(\text{mem8}[\text{address}])$  |
| <b>LDRSH</b> | Load a signed half-word into register     | $Rd \leftarrow \text{signExtend}(\text{mem16}[\text{address}])$ |

## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

- Load/Store một thanh ghi
- Ví dụ:

|      | addr     | Addr + 1 | Giá trị thanh ghi                   |
|------|----------|----------|-------------------------------------|
| Ldrb | 11001101 |          | 00000000 00000000 00000000 11001101 |
| Ldrh | 11001101 | 10100101 | 00000000 00000000 10100101 11001101 |

|       | addr     | Addr + 1 | Giá trị thanh ghi                   |
|-------|----------|----------|-------------------------------------|
| Ldrsb | 11001101 |          | 11111111 11111111 11111111 11001101 |
| Ldrsh | 11001101 | 10100101 | 11111111 11111111 10100101 11001101 |
| Ldrsb | 01001101 |          | 00000000 00000000 00000000 01001101 |
| Ldrsh | 11001101 | 00100101 | 00000000 00000000 00100101 11001101 |

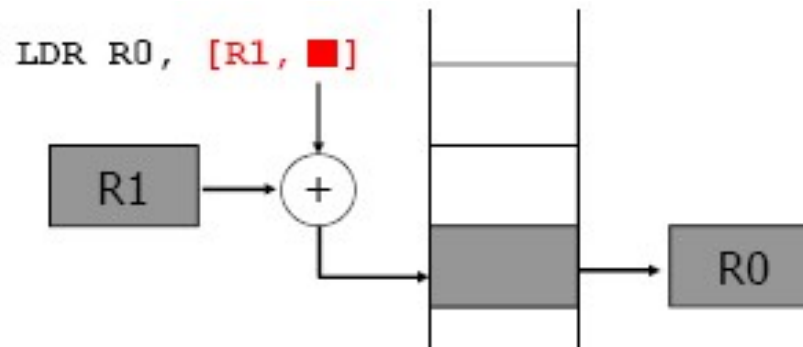
# CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

## ■ Load/Store một thanh ghi

- Chế độ địa chỉ Pre-indexed

- Địa chỉ bao gồm địa chỉ cơ sở (lưu trong thanh ghi) và các địa chỉ đoạn (offset)
- Mục đích: cho phép truy cập tới các vị trí ô nhớ trong một vùng nhớ.

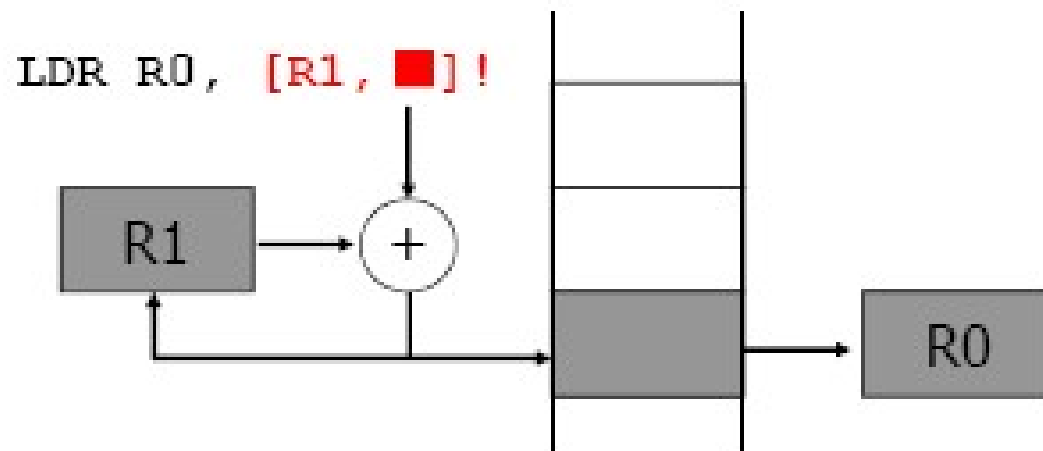
LDR R0, [R1, #4] ; R0 = mem[R1+4]  
; R1 không đổi



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

- Load/Store một thanh ghi
  - Chế độ địa chỉ Auto-indexing (Preindex with writeback)

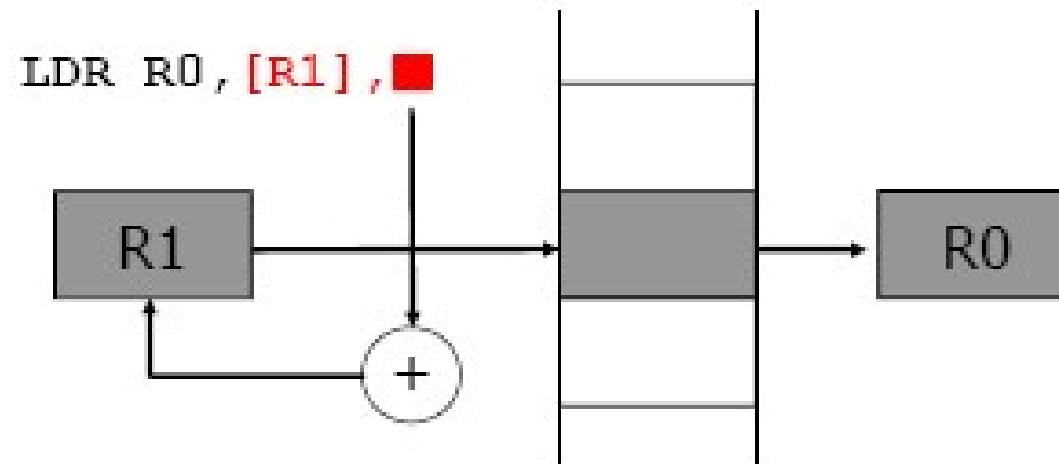
LDR R0, [R1,#4]! ; R0 = mem[R1+4]  
; R1 = R1 + 4



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

- Load/Store một thanh ghi
  - Chế độ địa chỉ Post-indexed
    - Không sử dụng dấu chấm than (!)

LDR R0, [R1], #4 ; R0 = mem[R1]  
; R1 = R1 + 4





## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

### ■ Load/Store một thanh ghi

- Lưu ý: Để nạp một số lớn hơn 255 vào thanh ghi:

LDR R1, =0x12345678

Nếu dùng lệnh MOV R1, #0x12345678, chương trình sẽ báo lỗi:

A1510E: Immediate 0x12345678 cannot be repre



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

### ■ Load/Store một thanh ghi

- Ví dụ:  $r0 = 0x00000000$

$r1 = 0x00000009$

$\text{mem32}[r1] = 0x01010101$

$\text{mem32}[r1+4] = 0x02020202$

Tìm giá trị của  $r0$ ,  $r1$  trong các trường hợp sau:

a. `LDR r0, [r1,#4]`

b. `LDR r0, [r1,#4]!`

c. `LDR r0, [r1],#4`





## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

### ■ Load/Store một thanh ghi

- Ví dụ: Tìm giá trị của R0, R1 trong các trường hợp sau:

- a. STR R0,[R1,#200]
- b. STR R0,[R1,R2,LSL #2]
- c. STR R0,[R1,#-16]!
- d. STR R0,[R1],#200



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

### ■ Load/Store một thanh ghi

- Ví dụ: Tính giá trị biểu thức

$$S = a1 * b1 + a2 * b2 + \dots + a10 * b10$$

trong đó địa chỉ ô nhớ chứa a1 lưu trong R8, b1 lưu trong R9.



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

### ■ Load/Store một thanh ghi

- Ví dụ: Tính giá trị biểu thức

```
MOV r11,#10
```

```
MOV r10,#0
```

```
LOOP
```

```
LDR r0,[r8],#4
```

```
LDR r1,[r9],#4
```

```
MLA r10,r0,r1,r10 ;  $r10 = r0 * r1 + r10$ 
```

```
SUBS r11,r11,#1
```

```
BNE LOOP
```

## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

### ■ Load/Store nhiều thanh ghi

- Cú pháp:

LDM|STM{cond}<mode> Rn{!},<register>

- Các chế độ địa chỉ

| Chế độ | Mô tả            | Địa chỉ bắt đầu | Địa chỉ kết thúc | Rn!        |
|--------|------------------|-----------------|------------------|------------|
| IA     | Increment after  | Rn              | $Rn + 4*N - 4$   | $Rn + 4*N$ |
| IB     | Increment before | $Rn + 4$        | $Rn + 4*N$       | $Rn + 4*N$ |
| DA     | Decrement after  | $Rn - 4*N + 4$  | Rn               | $Rn - 4*N$ |
| DB     | Decrement before | $Rn - 4*N$      | $Rn - 4$         | $Rn - 4*N$ |

## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

### ■ Load/Store nhiều thanh ghi

- LDMIA Rn, {<register>}
- Ví dụ:

```
LDMIA R0, {R1,R2,R3}
```

or

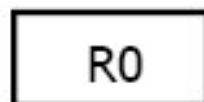
```
LDMIA R0, {R1-R3}
```

```
R1: 10
```

```
R2: 20
```

```
R3: 30
```

```
R0: 0x10
```



| addr  | data |
|-------|------|
| 0x010 | 10   |
| 0x014 | 20   |
| 0x018 | 30   |
| 0x01C | 40   |
| 0x020 | 50   |
| 0x024 | 60   |

## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

### ■ Load/Store nhiều thanh ghi

- LDMIA Rn!, {<register>}
- Ví dụ:

LDMIA R0!, {R1,R2,R3}

R1 : 10  
R2 : 20  
R3 : 30  
R0 : 0x01C

R0

| addr  | data |
|-------|------|
| 0x010 | 10   |
| 0x014 | 20   |
| 0x018 | 30   |
| 0x01C | 40   |
| 0x020 | 50   |
| 0x024 | 60   |

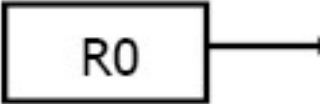
## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

### ■ Load/Store nhiều thanh ghi

- LDMIB Rn!, {<register>}
- Ví dụ:

LDMIB R0!, {R1,R2,R3}

R1: 20  
R2: 30  
R3: 40  
R0: 0x01C



| addr  | data |
|-------|------|
| 0x010 | 10   |
| 0x014 | 20   |
| 0x018 | 30   |
| 0x01C | 40   |
| 0x020 | 50   |
| 0x024 | 60   |

## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

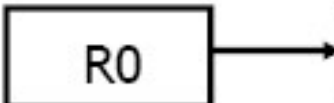
### ■ Load/Store nhiều thanh ghi

- LDMDA Rn!, {<register>}
- Ví dụ:

LDMDA R0!, {R1,R2,R3}

R1: 40  
R2: 50  
R3: 60  
R0: 0x018

| addr  | data |
|-------|------|
| 0x010 | 10   |
| 0x014 | 20   |
| 0x018 | 30   |
| 0x01C | 40   |
| 0x020 | 50   |
| 0x024 | 60   |



A diagram showing a box labeled 'R0' with an arrow pointing to the row in the memory table where the address is 0x018.



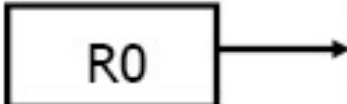
## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

### ■ Load/Store nhiều thanh ghi

- LDMDB Rn!, {<register>}
- Ví dụ:  
LDMDB R0!, {R1,R2,R3}

R1: 30  
R2: 40  
R3: 50  
R0: 0x018

| addr  | data |
|-------|------|
| 0x010 | 10   |
| 0x014 | 20   |
| 0x018 | 30   |
| 0x01C | 40   |
| 0x020 | 50   |
| 0x024 | 60   |



A diagram showing a box labeled 'R0' with an arrow pointing to the row in the memory table where the address is 0x018.



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

- Định nghĩa hằng số trong lập trình ARM
- DCB: Lệnh chỉ dẫn sử dụng một hoặc nhiều byte trong bộ nhớ để lưu dữ liệu
  - Cú pháp: Label DCB exp,{offset}
- Ví dụ:
  - Message1 DCB “Hello”
  - Value1 DCB 0x12 ; Phân bổ 1 byte cho ô nhớ Value1 = 12
  - Value2 DCW 0x1234 ; Phân bổ 2 byte cho Value2 = 1234
  - Value3 DCD 0x123456 ; Phân bổ 4 byte cho Value3
  - SQRT DCB 1,4,9,16; Phân bổ 4 byte cho 4 ô nhớ có giá trị lần lượt là 1, 4, 9 và 16



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

- Định nghĩa hằng số trong lập trình ARM
- Ví dụ: Tìm giá trị đảo của C123 và lưu vào ô nhớ ngoài



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

- Định nghĩa hằng số trong lập trình ARM
- Ví dụ: Tìm giá trị đảo của C123 và lưu vào ô nhớ ngoài

AREA vidu, CODE, READONLY

ENTRY

Main

LDR R1, Value

LDR R0, Result

MVN R1, R1

STR R1, [R0]

Value DCD 0xC123

Result DCD 0x0



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

- Định nghĩa hằng số trong lập trình ARM
- Ví dụ: Cộng hai số
  - Value1 = 12345678
  - Value2 = 23456789
  - Result = ?



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

- Giải thích các lệnh sau đây:

*MOV R0, R1,ROR #26*

*CMP R0,#256*

*MVNLT R0,#255*

*STR R0,[R1,R2,LSL#2]*

*STR R0,[R1,# -16]!*

*STRB R0,[R1,#1]!*

*STRB R0,[R1],R2*



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

- Căn chỉnh ô nhớ trong ARM
- Align {expr,{offset}} : Căn chỉnh ô nhớ hiện tại theo một kích thước nhất định.
  - Nếu expr không được chỉ ra, Align sẽ căn chỉnh vị trí lệnh từ nhớ tiếp theo.

## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

- Hoán chuyển dữ liệu giữa ô nhớ và thanh ghi

|                     |               |   |
|---------------------|---------------|---|
| SWP{B} Rd, Rm, [Rn] |               |   |
| SWP                 | WORD exchange | tmp = mem32[Rn]<br>mem32[Rn] = Rm<br>Rd = tmp |
| SWPB                | Byte exchange | tmp = mem8[Rn]<br>mem8[Rn] = Rm<br>Rd = tmp   |





## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

- Bài tập 1: Tính  $x = (\text{mem}[a] + \text{mem}[b]) - \text{mem}[c]$
- Bài tập 2: Tính  $x = (\text{mem}[a] + \text{mem}[b]) * \text{mem}[c]$
- Bài tập 3: Tính  $R1 = (R2 \ll 2) | (R3 \& 15)$

# CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

Bài tập 4: Giải thích chương trình sau

AREA Program, CODE, READONLY

ENTRY

Main

LDR R0, = Data1

EOR R1, R1, R1

LDR R2, Length

Loop

LDR R3, [R0]

ADD R1, R1, R3

ADD R0, R0, #4

SUBS R2, R2, #0x1

BNE Loop

STR R1, Result

SWI &11

Table AREA Data1, DATA  
DCW &0B1A

ALIGN

DCW &23C4

ALIGN

DCW &15E7

ALIGN

TablEnd DCD 0

AREA Data2, DATA

Length DCW (TablEnd - Table) / 4

ALIGN

Result DCW 0

END



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

- Bài tập 5: Viết chương trình thực hiện:

*Sao chép chuỗi “Hello world” từ một địa chỉ trong ô nhớ Srcstr sang một địa chỉ ô nhớ có tên Dststr.*

*Chương trình có sử dụng hàm con Funccopy để copy.*

- Bài tập 6: Viết chương trình thực hiện:

*So sánh hai chuỗi đã cho được lưu tại hai địa chỉ trong ô nhớ có tên là “String1” và “String2”. Lưu thông báo “Khác nhau” hoặc “Giống nhau” vào ô nhớ có tên Result*



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

```
        AREA    MYCODE, CODE, READONLY
        ENTRY

main
        ADR r1,srcstr
        ADR r0,dststr
        BL strcpy
srcstr DCB "Hello world",0
dststr DCB "Destination",0

Funcopy
        LDRB r2,[r1],#1
        STRB r2,[r0],#1
        CMP r2,#0
        BNE strcpy
        MOV pc,lr
        END      ;End of the program
```



# CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

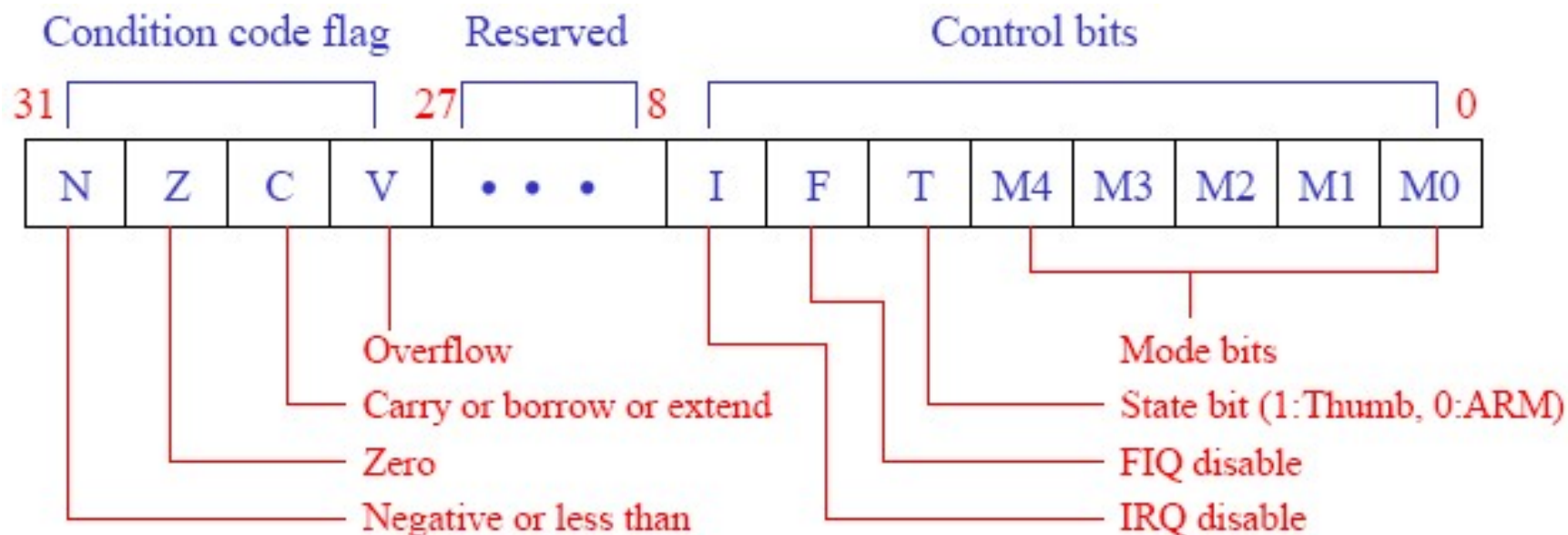
---

## NỘI DUNG

1. Giới thiệu về lập trình Assembly
2. Tập lệnh của ARM
  1. Lệnh xử lý dữ liệu (data processing)
  2. Điều khiển chương trình (flow control)
  3. Di chuyển dữ liệu (data transfer)
  4. Biệt lệ/Ngắt (Exception/Interrupt)

## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

### ■ Thanh ghi CPSR (Current Program Status Register)





## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

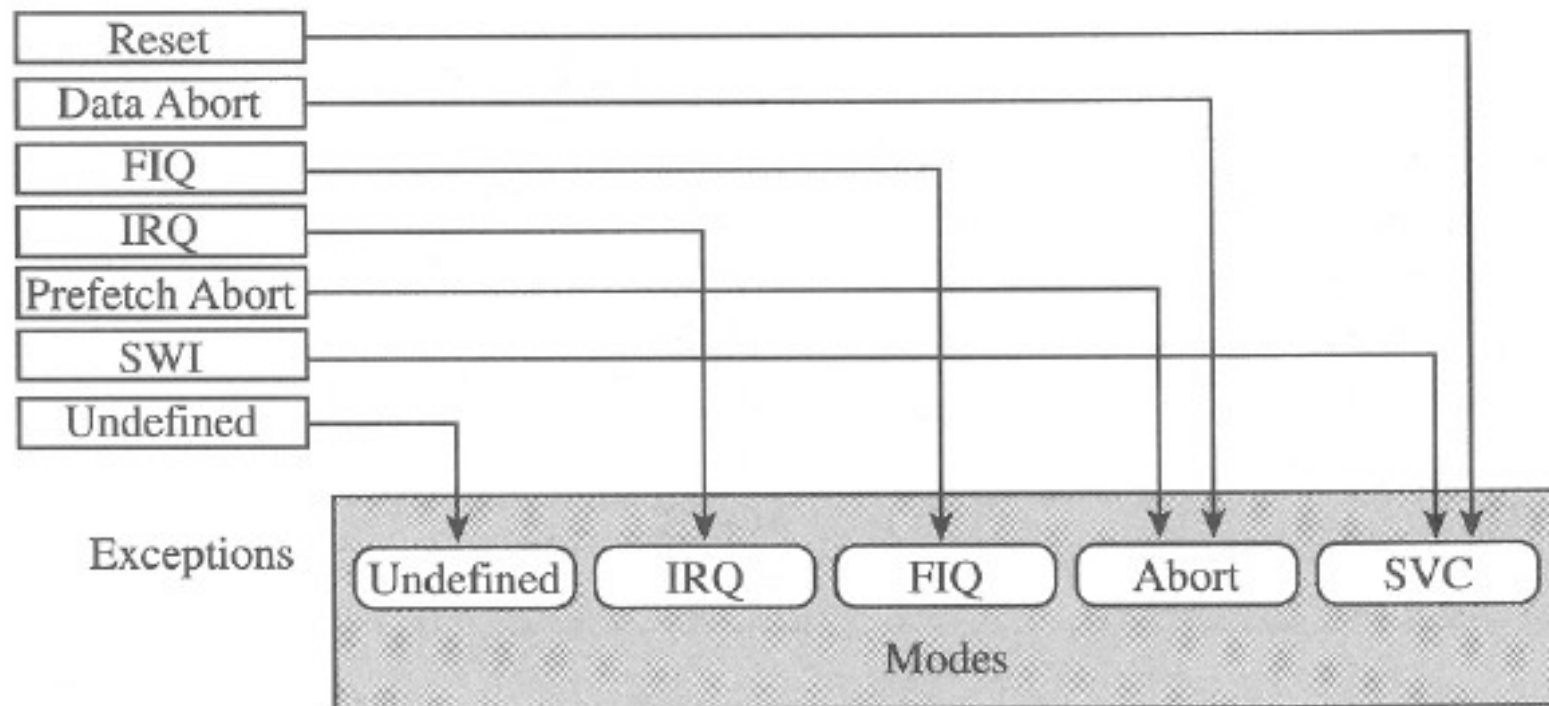
### ■ Thứ tự ưu tiên các biệt lệ

- Exception arise whenever the normal flow of a program has to be halted temporally.

| Priority | Exception                                   |
|----------|---|
| Highest  | Reset                                       |
|          | Data Abort                                  |
|          | FIQ   |
|          | IRQ   |
|          | Prefetch Abort                              |
| Lowest   | Undefined instruction or software interrupt |

# CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

## ■ Thứ tự ưu tiên các biệt lệ







## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

- Quá trình xử lý của CPU khi có biệt lệ:
  - Lưu giá trị CPSR vào SPSR tương ứng của chế độ tương ứng
  - Lưu PC vào LR của chế độ tương ứng
  - Thiết lập giá trị cho CPSR của chế độ tương ứng
  - Thiết lập giá trị cho PC để trở tới địa chỉ của trình xử lý biệt lệ (exception handler)

# CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

## ■ Ngắt mềm:

- Ngắt mềm (software interrupt) tạo ra một biệt lệ cho phép ứng dụng gọi một số tác vụ của hệ điều hành.
- Cú pháp: SWI{cond} SWI\_number

SWI{<Cond>} SWI\_number

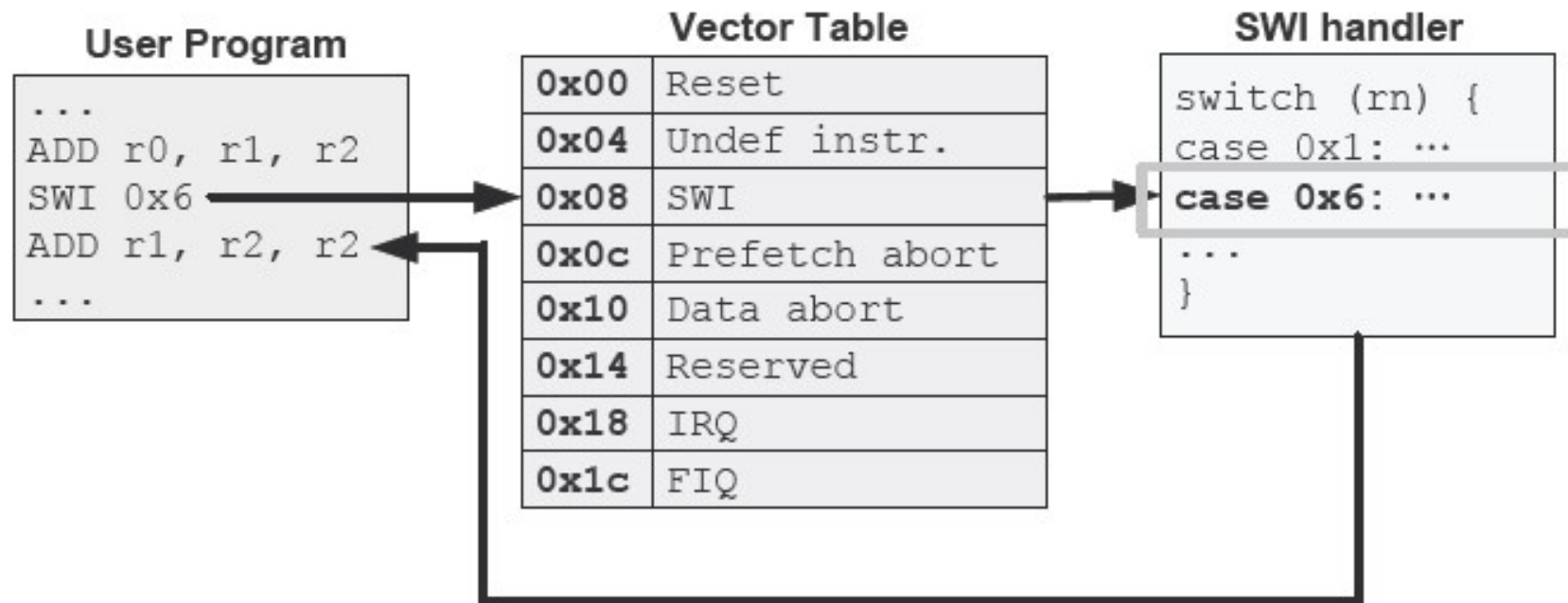
- lr\_svc = address of next instruction following SWI
- SPSR\_svc = CPSR
- pc = 0x08
- CPSR mode = SVC
- CPSR I = 1 (mask IRQ interrupts)

After SWI routine, return to the calling program by

MOV pc, lr\_svc

# CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

## ■ Ngắt mềm:



# CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

| Số hiệu ngắt | Mô tả chức năng  | Đầu vào  | Đầu ra  |
|--------------|--|--|---|
| SWI 0x00     | Hiển thị một ký tự   | R0 lưu ký tự   | Hiển thị trên màn hình ký tự                                    |
| SWI 0x02     | Hiển thị một chuỗi ký tự ra màn hình   | R0 lưu địa chỉ của chuỗi ký tự                                   | Hiển thị trên màn hình chuỗi ký tự                              |
| SWI 0x11     | Dừng chương trình  |  |   |
| SWI 0x12     | Đăng ký sử dụng vùng nhớ trên Heap   | R0 lưu kích thước (đơn vị là byte)                               | R0: địa chỉ của vùng nhớ  |
| SWI 0x13     | Giải phóng vùng nhớ trên Heap  |  |   |
| SWI 0x66     | Mở một file dữ liệu. Chế độ mở được lưu trên R1: <ul style="list-style-type: none"> <li>- 0: Mở để đọc</li> <li>- 1: Mở để ghi</li> <li>- 2: Mở để ghi tiếp</li> </ul> | R0: lưu tên file<br>R1: lưu chế độ mở file                       | R0: Kết quả quá trình mở file. Nếu file không mở được, R0 = -1. |
| SWI 0x68     | Đóng file  | R0: lưu tên file   |   |
| SWI 0x69     | Viết ghi chuỗi ký ra file hoặc ra màn hình   | R0: tên file hoặc Stdout<br>R1: Địa chỉ chuỗi ký tự              |   |
| SWI 0x6a     | Đọc một chuỗi ký tự từ một file  | R0: tên file<br>R1: địa chỉ đích<br>R2: số byte lớn nhất cần đọc | R0: số byte được lưu trữ  |



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

### ■ Ngắt mềm:

Ví dụ 1: Viết chương trình hiển thị chuỗi ký tự “Hello world” ra màn hình.

```
.equ swi_stdout,0x02      ; gán số hiệu ngắt với tên ngắt  
ldr r0,=TextString      ; nạp địa chỉ chuỗi ký tự vào thanh ghi  
swi swi_stdout           ; gọi ngắt  
TextString: .asciz "Hello world\n"
```



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

### ■ Ngắt mềm:

Ví dụ 2: Mở file và ghi chuỗi ký tự vào file.

```
InFileName:    .asciz "Infile1.txt"
Message:       .asciz "Hello there \n"
.equ SWI_Open,0x66
ldr    r0,=InFileName
mov    r1,#2           @ input mode
swi    SWI_Open
ldr    r1,=Message
swi    0x69
swi    0x68
```



## CHƯƠNG 3-LẬP TRÌNH HỢP NGỮ ARM

---

- Bài tập 1: Viết chương trình thực hiện:

*Sao chép chuỗi “Hello world” từ một địa chỉ trong ô nhớ Srcstr sang một địa chỉ ô nhớ có tên Dststr. Sau đó hiển thị chuỗi ký tự từ ô nhớ Dststr.*