Pontificia Universidad
JAVERIANA
Cali

Res. 2333 del 2012

Vigilada Mineducación Resolución 12220 de 2016

ARQUITECTURA DE COMPUTADORES II: LABORATORIO.

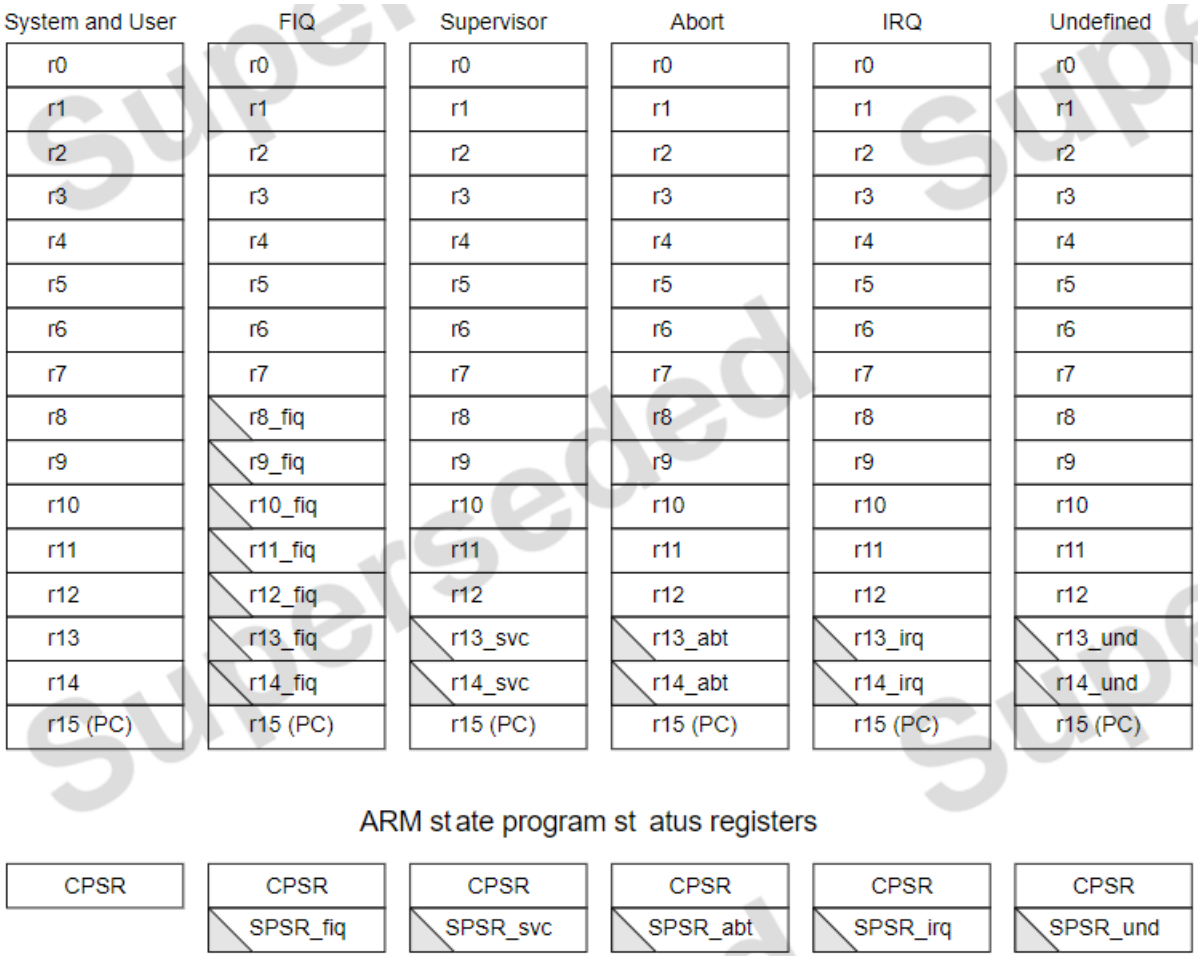
Session 2 – August 3

Session 2: Outline

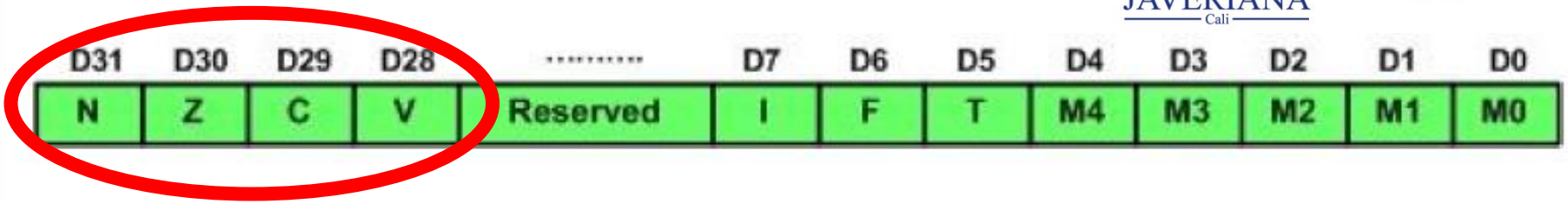
- ARM Register set
- CPSR
- Processor modes
- LDR Instruction
- ADD and SUB Instructions

ARM Register Set

- In privileged modes, another register, the *Saved Program Status Register* (SPSR), is accessible.
- In ARM state, bits [1:0] of r15 are zero

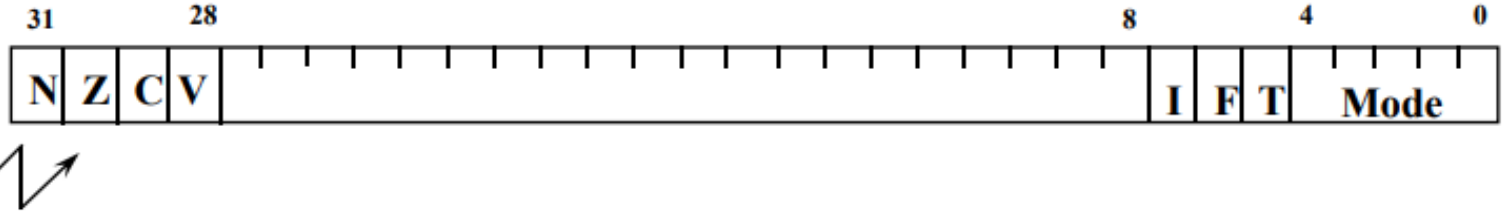


ARM CPSR (Current Program Status Register



- El CPSR es el registro que contiene las banderas que indican condiciones aritméticas.
- Es un registro de 32 bits.
- BANDERAS CONDICIONALES: Indican alguna condición se origina de la ejecución de una instrucción.
- C (ACARREO): Esta bandera se activa después de una suma o resta de 32 bits, usado para detectar error en operaciones sin signo.

The Program Status Registers (CPSR and SPSRs)



Copies of the ALU status flags (latched if the instruction has the "S" bit set).

* **Condition Code Flags**

  N = **N**egative result from ALU flag.
  Z = **Z**ero result from ALU flag.
  C = ALU operation **C**arried out
  V = ALU operation o**V**erflowed

* **Mode Bits**

  M[4:0] define the processor mode.

* **Interrupt Disable bits.**

  I = 1, disables the IRQ.
  F = 1, disables the FIQ.

* **T Bit (Architecture v4T only)**

  T = 0, Processor in ARM state
  T = 1, Processor in Thumb state

Processor Modes

The Mode Bits

xPSR[4:0]	Mode
10000	User mode
10001	FIQ mode
10010	IRQ mode
10011	Supervisor mode
10111	Abort mode
11011	Undefined mode
11111	System mode


```
1  MOV R1,#0x0000009C
2  MOV R2,#0xFFFFFF64
3  ADDS R2,R1,R2
```

- Si se necesita que una instrucción **actualice el valor del CPSR**, se coloca S al final de mnemotécnico.

- Muestre el estado de las banderas C y Z en el siguiente código.
- C=1 porque hay un acarreo más allá del bit 31.
- Z=1 debido a que el resultado R2 tiene valor de 0 después de la operación.

0x0000009C

\+

0xFFFFFF64

0x100000000

LDR Instruction

Is used to load data to a register from memory.

LDR load 32 bits (word)

LDRB load 1 byte

LDRH Load 16 bits (Half word)

LDRS load signed byte

LDRSB Load sign extension

LDRSH Load Signed half word

LDM Load multiple words

```
LDR Rd,=Value
```

ADD


```
1          AREA SUMA_SIMPLE, CODE, READONLY
2      ENTRY
3          MOV R0, #0X25
4          MOV R1, #0X23
5          ADD R2, R1, R0
6  HERE    B   HERE
7      END
```

ADD Rd,Rn,Op2 ;ADD Rn to Op2 and store the result in Rd

;Op2 can be Immediate value #K (K is between 0 and 255)

;or Register Rm

```
ADC R0, R1, R2;   ;R0 = R1+R2 +C Add with carry C is carry bit.
```

ADD INSTRUCTION

- Los primeros bits son el campo de condición.
- Bits 27 y 26 están siempre en 0 en una instrucción ADD.
- Bit 25 define el tipo del segundo operando. I=1 es un valor inmediato, de otra manera será un registro.
- Bits 24 a 21 son el código de operación de la instrucción ADD.
- Bit 20 define si la instrucción debe actualizar los flags o no. En una instrucción ADD este bit es cero mientras que en ADDS es uno.
- Bit 19 a 16 define el primer operando, puede ser un registro R0 a R12.
- Bit 15 a 12 definen el registro de destino.
- Bits 11 a 0 definen el segundo operando ($1 - 255_{10}$).

EJEMPLO

Sume los números 0x25 y 0x34.

```
1           AREA SUMA_SIMPLE, CODE, READONLY
2       ENTRY
3           MOV R0, #0X25
4           MOV R1, #0X34
5           ADD R2, R1, R0
6   HERE    B   HERE
7       END
```

```
            MOV R0, #0X25
            ADD R2,R0, #0X34
HERE        B   HERE
```

EJEMPLO ADC

Se utiliza para sumar variables de 64 bits

0XA A770 0C00 = 45.758.811.136
0XB 63D8 3B71 = 48.919.755.633
TOTAL = 0X16 0B48 4771

```
LDR R0,=0XA7700C00
LDR R1, =0XA
LDR R2,=0X63D83B71
LDR R3,=0XB
ADDS R4,R0,R2
ADC R5,R1,R3
```

SUB Instruction


```
SUB     Rd,Rn,Op2       ;Rd=Rn – Op2
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam accumsan ligula ut gravida mattis.

```
2       ENTRY
3           LDR R0,=0X23
4           SUB R0,#0X23
5HERE       B   HERE
6       END
```