**ARM**

El primer ARM fue desarrollado por Acorn RISC Machine entre 1983 y 1985, ARM Limited solo se dedica al desarrollo y diseño de procesadores RISC, no los hace, cubre el 75% de mercado, concede licencias de diseño.

Los microprocesadores RISC de 37 registros e instrucciones de 32 bits (17 o 18 visibles), al principio utilizaban bus tipo Von Newman (ARM7), posteriormente bus tipo Harvard, con tipos de datos de 8/16/32 bits y 6 o 7 modos de operación, cuenta con una estructura simple de alta velocidad y bajo consumo de potencia, con transferencias de memoria/registros, Load/Store.

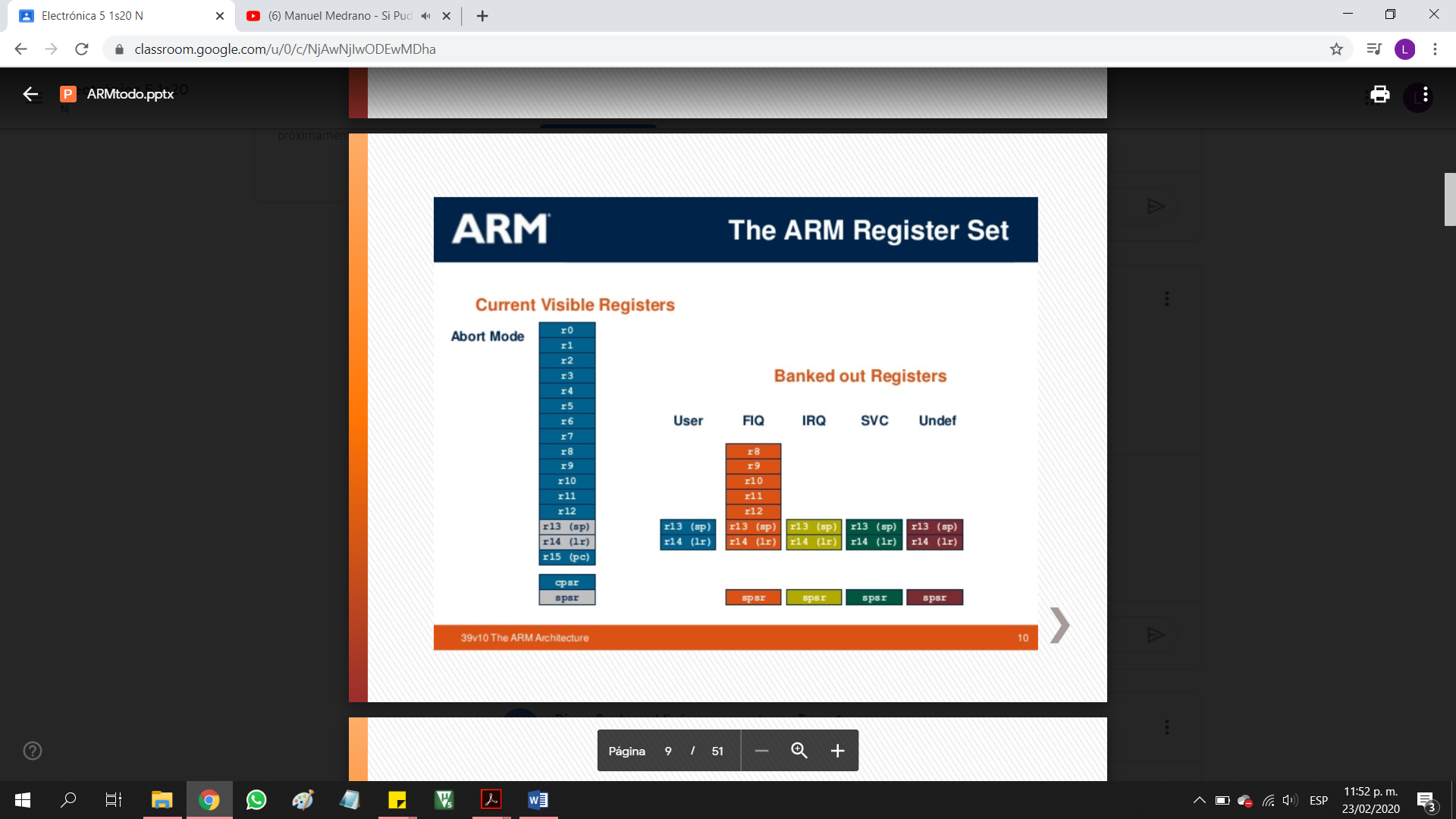
Periféricos integrados: UART, GPIO, LCD controller etc.

* **Esquema interno**
  + **Register Bank:** 2 puertos de lectura más uno de escritura para acceso a todos los registros. Un puerto de lectura más uno de escritura para acceder a R15.
  + **Barrel Shifter:** desplaza o rota el contenido de los registros.
  + **Adrress register mas incrementer:** almacena las direcciones de acceso a memoria.
  + **Data register:** almacena datos en las transferencias con memoria.
  + **Instruction decode & control.**

**Pipeline:**

* **Fetch:** se obtiene la instrucción de memoria y se almacena en el pipeline de instrucciones.
* **Decode:** se decodifica la instrucción y se activan las señales de habilitación Datapath.
* **Execute:** operaciones y transferencias de datos ocupando el Datapath.

Todas las instrucciones ocupan al datapath en un ciclo, que activa las líneas de decodificación, durante el primer ciclo de datapath se obtiene la siguiente instrucción. Una instrucción de salto provoca el borrado del pipeline, en el tercer ciclo de la instrucción se carga la segunda instrucción posterior.

**Modos de funcionamiento:**

* **User (USR):** estado normal de ejecución de programas.
* **FIQ:** empleado en transferencias de alta velocidad.
* **Supervisor (SVC):** modo protegido para el sistema operativo.
* **Abort Mode (ABT):** usado cuando se aborta el ciclo fetch de datos o instrucciones.
* **IRQ:** interrupciones de propósito general.
* **Underline (UND):** emulación de software.
* **System (SYS):** para ejecutar tareas del Sistema operativo en modo privilegiado.

**Registros:**

***Registros de propósito general:***

* Unbanked registers R0-R7: son comunes a todos los modos de funcionamiento.
* Banked registers R8-R14: se accede a distintos registros dependiente del modo en que se encuentre. R8-R12, hay dos banked physical register. En R13-R14 hay seis banked physical register.

***R13 (Stack pointer | Puntero de pila)[SP]:*** no tiene instrucciones específicas, exceptuando en modo Thumb, maneja la pila, cada modo de funcionamiento tiene su propio SP que debe ser inicializado a una zona de memoria dedicada.

***R14 (Link Register)[LR]:*** hay uno para cada modo de direccionamiento, almacena la dirección de retorno en una llamada a una rutina, evitando el almacenamiento del contador de programa en la pila, proporcionando un retorno rápido, también almacena la dirección de retorno cuando se producen excepciones.

***R15 (Program Counter)[PC]:*** contiene la dirección de instrucción que se va a ejecutar, si se puede emplear como cualquier otro registro de P-G. La lectura mediante una instrucción devuelve el valor de dirección actual mas 8 bytes. La escritura provoca un salto a la dirección cargada.

***Registro de estado CPSR(Current Program Status Regsiter):*** es accesible en todos los modos de funcionamiento y contiene las banderas de condición.

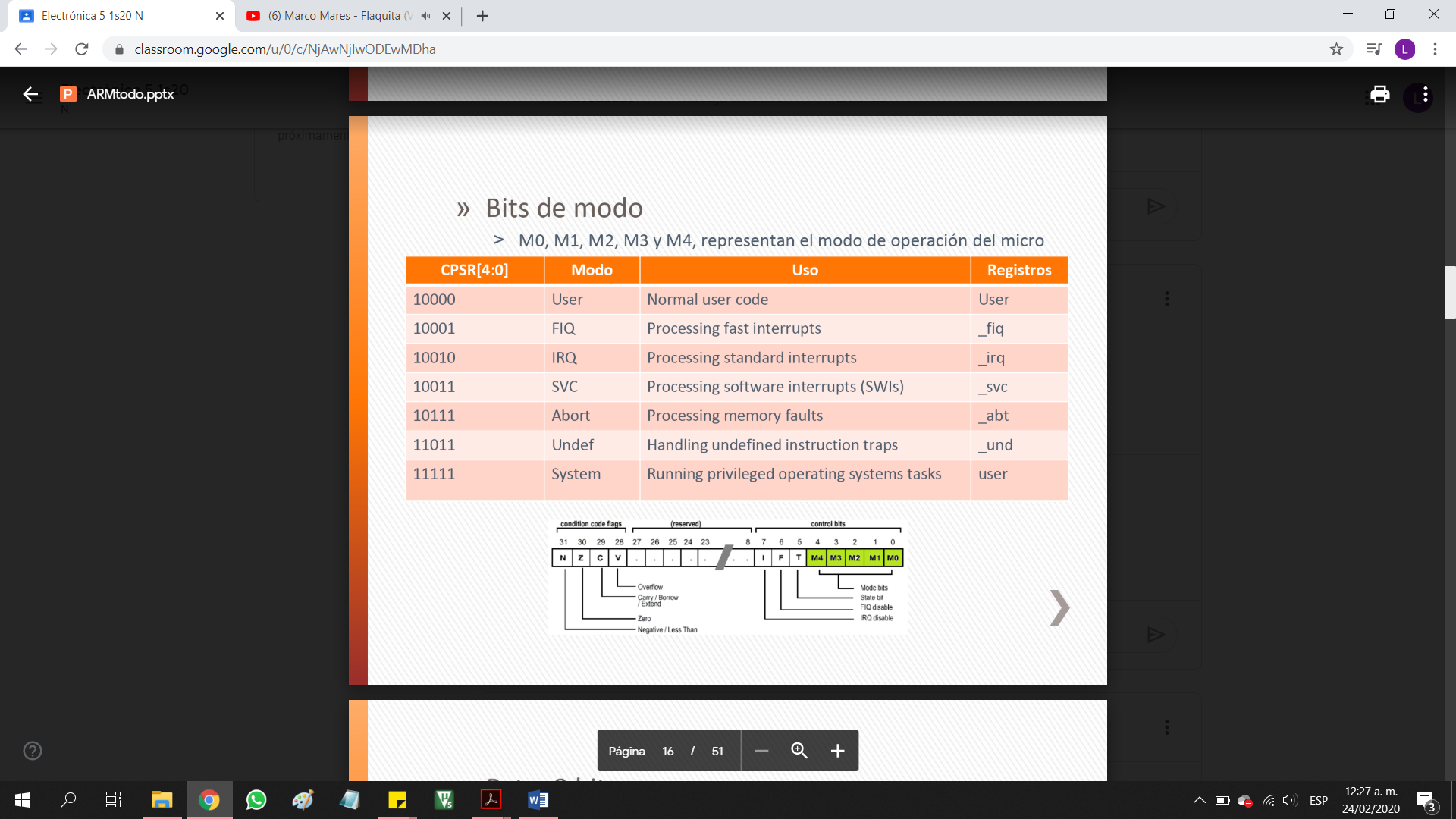
***SPSR(Saved Program Status Register):*** contiene una copia del CPSR cuando se entra a modo privilegiado, todos los modos lo tienen excepto user y system.

**Banderas de Condiciones:** N, Z, C, V, Q.

**Bits de Control:** se modifican durante la ejecución de una excepción o por programa.

* I o F, puesto a 1, deshabilita interrupciones IRQ o FIQ, respectivamente.
* T, puesto a 0, ejecución ARM | T, puesto a 1, ejecución Thumb

**Bit de Modo:**



**Tipos de datos:** byte-8bits, halfword-16bits, Word-32bits.

**Organización de Datos en Memoria ARM:** little endian y big endian.

**Esto ya no**

**Instrucciones de Procesos de Datos**

Permite realizar operaciones aritméticas y lógicas sobre el contenido de los registros. Los operandos son registros o valores inmediatos. **ADD R0,R1,R2** ; R0 operando destino, R1 y R2 operandos fuente.

***Instrucciones Aritméticas:***

* **Suma binaria:** ADD R0,R1,R2 ;R0=R1+R2
* **Suma con acarreo:**AD**C** R0,R1,R2 ;R0=R1+R2+C
* **Resta binaria:** SUB R0,R1,R2 ;R0=R1-R2
* **Resta con acarreo:** SBC R0,R1,R2 ;R0=R2-R1+C-1
* **Resta binaria inversa:** RSB R0,R1,R2 ;R0=R2-R1
* **Resta binaria con acarreo:**RSC R0,R1,R2 ;R0=R2-R1+C-1

***Instrucciones Lógicas:***

* **Producto lógico:** AND R0,R1,R2 ;R0=R1 and R2
* **Suma lógica:**ORRR0,R1,R2 ;R0=R1 or R2
* EOR R0,R1,R2 ;R0=R1 xor R2
* BIC R0,R1,R2 ;R0=R2 and not R1
* **Suma exclusiva**
* **Borrador de un bit**

***Instrucciones de Movimiento de Datos***

Estas se realizan entre registros, son consideradas de proceso de datos, porque los modifican. Omiten al primer operando fuente. Copian el segundo operando de forma directa o en complemento a uno.

* **MOV R0,R2** ; R0=R2
* **MVN R0,R2** ; R0=not R2

***Instrucciones de Transferencia de Datos***

Estas se realizan con memoria.

***Instrucciones de Comparación***

No generan resultado, solo posicionan los códigos de condición [cc] del registro estado

* **Compare:** CMP R1,R2 ; posiciona los cc en función del resultado de operación(R1-R2)
* **Compare Negated:** CMN R1,R2 ; “ “ “” “” “”(R1+R2)
* **Test:** TST R1,R2 ; “” “” “” “”(R1&R2)
* **Test Equal:** TEQ R1,R2 ; “” “” “” “”(R1^R2)

***Operandos Inmediatos***

Limitados por (0…255)\*22n 0<=n<=12

* **ADD R0,R1,#5** ; R0=R1+5, el valor inmediato va precedido de #
* **ADD R3,R3,#1** ; R3++
* **ADD R8,R7,#&FF** ; R8=R7&0Xff, & significa hexadecimal

***Instrucciones de desplazamiento y rotación***

Pueden modificar un operando antes de realizar las operaciones anteriores, le toma un ciclo de reloj ejecutar la instrucción.

**ADD R1,R2,R3,LSL#5** ; R1=R2+R3\*32 | R3 se desplaza 5 veces a la izquierda, lo que equivale a añadirle 5 ceros a la derecha. 2^5=32.

Es posible indicar el número de desplazamiento mediante un registro.