TEORICO ELECTRONICA DIGITAL 3 SEGUNDO PARCIAL

En el siguiente resumen están todos los teóricos encontrados en parciales de digitales 3, como siempre por ahí mejor corroborar algunas cosas y no dar por sentado lo que este escrito acá, sin embargo la mayoría de las respuestas fueron sacadas del manual del micro y los códigos simulados en LPCXpresso para corroborar los resultados.

1) *Mostrar cómo se mueven los bits de los registros en operaciones de movimiento.*

ASR#3

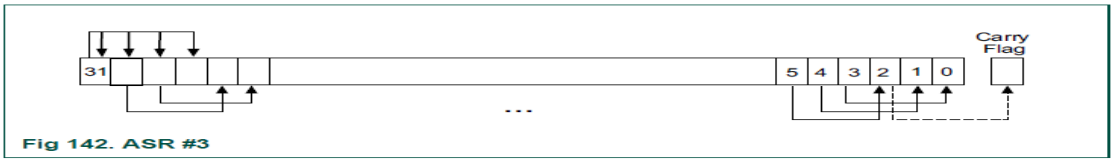


Fig 142. ASR #3

LSR#3

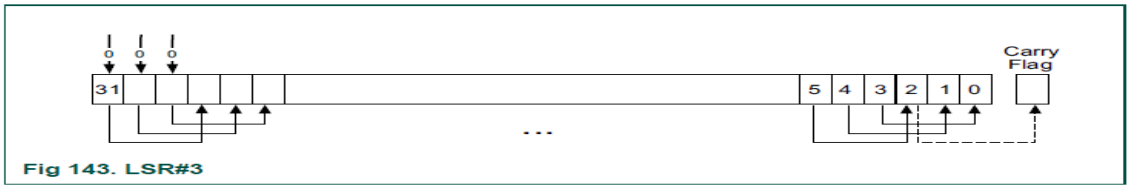


Fig 143. LSR#3

LSL#3

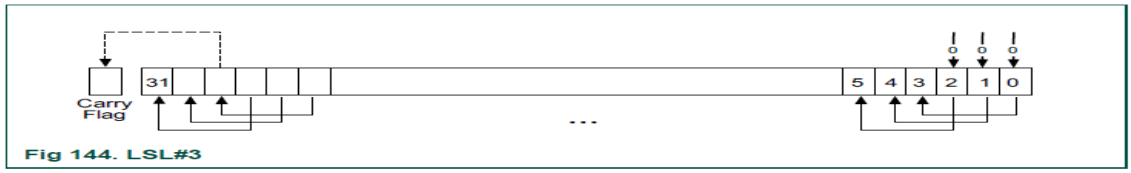


Fig 144. LSL#3

ROR#3

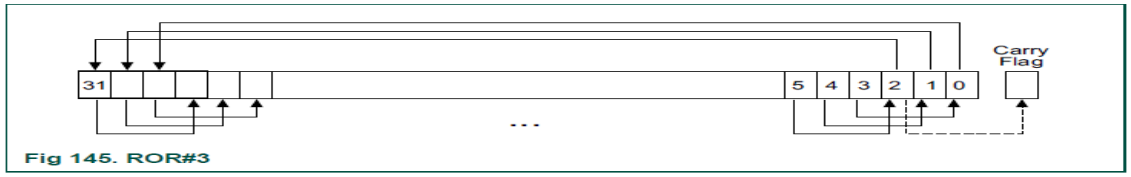


Fig 145. ROR#3

RRX

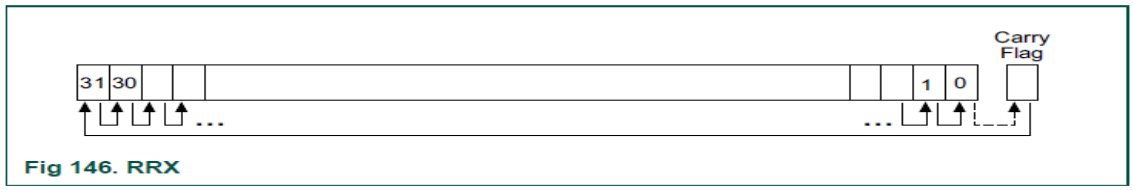


Fig 146. RRX

2) *Resuelva la siguiente operación e indique el valor de las banderas de estado :*

r0=0x0000000A
r1=0xFFFFFFFF

ANDS r0,r0,r1

Considerando solo los últimos 8 bits de los registros se resuelve la operación AND quedando el siguiente resultado:

1010	A
x	
1111	F
1010	A

En este caso las banderas de estado no se ven afectadas.
Breve explicación de cuáles son las banderas y que las hace posicionarse:

n(Negative or less than flag) : se pone en "1" cuando el resultado de la operación dio negativo, de lo contrario se mantiene en "0".

z(Zero flag): se pone en "1" cuando el resultado de la operación dio cero, de lo contrario se mantiene en "0".

c(Carry or borrow flag): se pone en "1" cuando la operación genera un carry, de lo contrario se mantiene en "0". **un carry ocurre cuando el resultado de una suma es mayor a 2^32, si el resultado de una resta es positivo o cero o como resultado de una operación de desplazamiento o una operación lógica**

v(Overflow flag): se pone en "1" cuando la operación causa un overflow, de lo contrario se mantiene en "0".**un overflow ocurre cuando el resultado de una suma, resta o comparación es mayor o igual que 2^31 o menor que -2^31**

q (Sticky saturation flag): *= 0* indica que la saturación no se ha producido desde el reset o desde que el bit fue llevado a cero por última vez.***1* =** indica cuando un SSAT o USAT llevan a una saturación.
Este bit se pone a cero por software mediante una instrucción MRS.

3) *Determine el valor de r0 en cada instrucción por separado, teniendo en cuenta los valores iniciales en cada instrucción.*

r0=0x000000F0 = 0b11110000
r1=0x00000010 = 0b00010000
r2=0x00000011 = 0b00010001

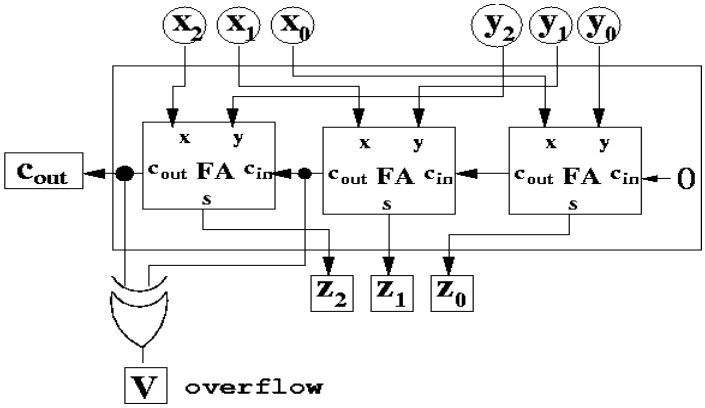
ANDS r0,r1,r2 LSL#1 ***//r0 = 0x00***
ADD r0,r0,r0 LSL#2 ***//r0 = 0011110000 + 1111000000 = 10010110000 = 0x4B0***
SUB r0,r0,r0 ASR#2 ***//r0 = 0xB4***

```
MVN     r0,r1           //r0 = 0b11101111 = 0xFFFFFFEF
STR     r0,[r0,#-200]!  //r0 = 0xF0 - 0xC8 = 0x28 (no se sabe con certeza)
BIC     r0,r0,#0x20     //r0 = 0xD0
STR     r0,[r1,r2,LSL#3] //r0 -> r1 + 0x10000 (no lo entendi, no veo q
cumpla ningun formato) segun march se guarda r0 en r1+(r2<<3)
MOV     r0,r0 LSL#2     //r0 = 0x3C0
MVN     r0,#0           //r0 = 0xFFFFFFFF
```

4) *¿Qué indica la bandera de overflow? Indique la ecuación o circuito lógico para detectarlo.*

La bandera de overflow (v) indica cuando el resultado de una suma, resta o comparación es mayor o igual que 2^31 o menor que -2^31.

A continuación el circuito para detectar el overflow:



5) *¿Qué sucede cuando se produce una excepción en ARM?*

Ante una excepción, el procesador suspende su actividad y realiza la siguiente serie de operaciones: **0.**Finaliza la ejecución de la instrucción en curso.**1.**Guardar los registros R0 a R3, R12, LR, PSR y PC en el stack. Esto ocupa el bus SYSTEM **2.**Leer el vector de interrupciones en la posición indicada por la dirección base de éste y el número de excepción, mediante el bus ICODE. Dado que los pasos “1” y “2” ocurren en espacios separados, se realizan simultáneamente. **3.** Actualizar el PC **4.**Llenar la pipeline, también en paralelo con el paso “1”, al ocurrir los accesos a memoria en buses separados. **5.** Colocar un valor especial: EXC_RETURN en el LR.

Terminado el procesamiento de la excepción, el procesador se encontrará con una instrucción de retorno de función, por lo que copiará el contenido de LR al PC. El valor especial cargado al iniciar la excepción en el LR, al ser visto en el PC se lee como “esto es un retorno de excepción”, por lo que el procesador inicia la secuencia correspondiente, que consta de los siguientes pasos: **1.**Recuperar los ocho registros del stack **2.**Observar si debe volverse a otra excepción o al programa principal **3.**Elegir el stack correspondiente y actualizar el SP. Si retorna a una excepción, utilizará MSP. Si retorna al programa principal (modo THREAD), será el MSP o el PSP según cómo esté configurado. **Extraído de del SASE 2013 para cortexM3**

6) *Las instrucciones de proceso de datos tienen la forma:*

ADD{cond}{S} <dest>,<lhs>,<rhs>

Explique las distintas formas que puede tener el operando <rhs> y de un ejemplo en cada caso.

<rhs> puede ser un registro, un valor inmediato, un registro rotado o un registro desplazado. A continuación se proporciona un ejemplo para cada uno de los casos mencionados:

ADD R0,R0,R1 //←<rhs> es un registro

ADD R0,R0,#2 //←<rhs> es un valor inmediato

ADD R0,R0,R1,LSL#2 //←<rhs> es un registro desplazado

ADD R0,R0,R1,ROR#2 //←<rhs> es un registro rotado.

7) *Explique la diferencia entre las siguientes dos instrucciones en lenguaje ensamblador* ***(SABERSE ESTA SI O SI LA NOMBRO UNAS 23987129 VECES):***

LDR R0,=Tiempo

LDR R0,Tiempo

Al utilizar la instrucción LDR R0,=Tiempo lo que se hace es cargar en R0 la dirección de donde está ubicada la etiqueta Tiempo. Debido a que dicha dirección no existe físicamente en memoria lo que hace el compilador es crear en algún lugar de memoria una constante de 4bytes con la dirección de la etiqueta, entonces el compilador suma al PC el valor necesario para llegar a donde se encuentra creada la constante con la dirección y carga ese valor en R0. Este modo de direccionamiento se llama “PC Relative Adressing”, es un modo especial de direccionamiento preindexado, mediante el cual el ensamblador calcula automáticamente el Offset necesario.

En caso de usar LDR R0,Tiempo lo que sucederá es que se cargara en R0 el dato que está almacenado en el lugar de la etiqueta.

Por ejemplo en el caso que tengamos una tabla como la siguiente:

```
tabla7seg:   .word  0x07018004          //0
             .word  0x06000000          //1
             .word  0x05030004          //2
```

Se debe utilizar LDR R0,=tabla7seg porque de lo contrario, si se usara sin el igual, en vez de ir a la tabla y desde ese momento elegir el valor inmediato a traer para representar el nro en un display 7 segmentos lo que se hará será asignarle a R0 el valor 0x07018004 que no es lo deseado. **Se sugiere consultar para tener claro este tema**

8) *Tipos de variables en C*

Las más comunes son las siguientes:

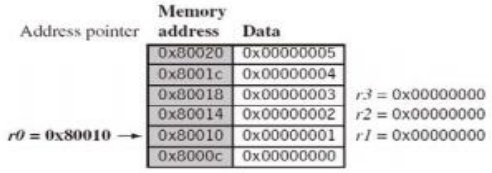
TIPO DE DATOS	SE ESCRIBE	MEMORIA REQUERIDA*	RANGO ORIENTATIVO*	EQUIVALENCIA EN PSEUDOCÓDIGO	OBSERVACIONES
Entero	int	2 bytes	- 32768 a 32767	Entero	Uso en contadores, control de bucles etc.
Entero largo	long	4 bytes	- 2147483648 a 2147483647	Entero	Igual que int pero admite un rango más amplio
Decimal simple	float	4 bytes	$-3{,}4\cdot10^{38}$ a $3{,}4\cdot10^{38}$	Real	Hasta 6 decimales. También admite enteros
Decimal doble	double	8 bytes	$-1{,}79\cdot10^{308}$ a $1{,}79\cdot10^{308}$	Real	Hasta 14 decimales. También admite enteros
Carácter	char	1 bytes	0 a 255	Alfanumérica	Carácter, independiente o parte de una cadena

* Podría variar

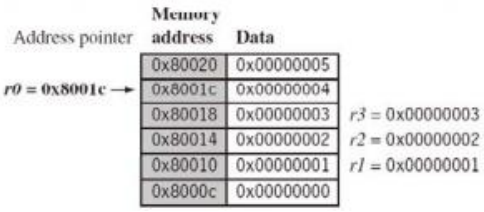
9) *¿Qué es y para qué sirve el Link Register?*

El link Register (R14) se encarga de guardar la dirección mediante la cual se vuelve de una subrutina al programa principal.

10) *Con estas condiciones iniciales:*



Luego de ejecutar LDMIA R0!,{R1-R3} ¿que contienen los regs R0,R1,R2 y R3?Rta:



11) *¿Qué es y cómo funciona el MAM?*

El MAM (Memory Acelerator Module) es un chip de aceleración de memoria. Es útil para mejorar el rendimiento del procesador ARM cuando se corre un programa desde la memoria flash. Se configura mediante un cuadro de dialogo dentro del cual se encuentran:

- MAM control group: -MAMCMR: controla el modo de funcionamiento.

  -Mode: muestra cual es el modo elegido.

- MAM timming group: -MAMTIM: setea el CCLK para cada ciclo de recuperación MAM.

  -Fetch: muestra el seteo de tiempo.

Esta pregunta fue encontrada en un parcial muy viejo donde incluso no se usaba la misma LPC asi que si hay poco tiempo mejor obviarla

Otros datos teóricos que pueden llegar a servir

-ARM procesa datos solo en registros.

-Los consumos estan atados a la frecuencia de trabajo, a mayor frecuencia mayor consumo.

-Se usa ensamblador Thumb2 el cual mejora al Thumb, tiene instrucciones de 16 y 32 bits lo cual permite poca memoria para almacenar casi lo mismo.

-Direccionamiento **Preindexado: STR{cond}<srce>,[<base> {,<offset>}].** Donde <srce> es el registro que contiene el dato a trasferir, <base> el registro que contiene la dirección de memoria y <offset> es un nro opcional que es adicionado a la dirección antes de que el dato sea almacenado.

<srce>=<base> +<offset>. Ejm: STR R0,[R1,R2,LSL#2] //R1+R2*4

-Write back: frecuentemente una direccion debe ser actualizada en el registro base, para ello se agrega "!" al final, el registro se actualiza al valor <base>+<offset>. Ejm: STR R0,[R1,#-16]!, esto almacena R0 en la dirección R1-16 y luego ejecuta automáticamente SUB R1,R1,#16.

-Direccionamiento **Postindexado: STR{cond}<srce>,[<base>],<offset>.** En este modo el offset no es agregado hasta la sig. Instrucción.

-Direccionamiento **PC Relative addressing:** forma especial de preindexado

LDR <dest>,<expression>. <expression> contiene una dirección y en este caso la instrucción usara R15 como registro base (PC). El assembler calculara el offset automáticamente. Si el offset no está en el rango (-4095 a +4095) se produce un error. Ejm LDR R0,default. default es una etiqueta en el programa y su valor será la diferencia de la ubicación entre esta y el PC.

-Existen instrucciones para hacer multiples Loads y Stores, LDM y STM respectivamente.

-Si se pasan más de 4 parámetros a una rutina de assembler el ensamblador los acomoda en el stack.

-La sentencia .w es utilizada por el ensamblador para usar la instrucción larga (32 bits) debido a que existen instrucciones que hacen lo mismo pero unas de 16 bits y otras de 32bits.