E.P.S. – U.A.M. – Arquitectura de Ordenadores – 25/Enero/2021

Apellidos, Nombre	DNI	Test (3p)	C1 (1p)	C2 (1p)	P1 (2.5p)	P2 (2.5p)	TOTAL

Cuestión 1.- Se rediseña el set de instrucciones (ISA) de un procesador y se debe evaluar el efecto en el rendimiento. Las instrucciones obtenidas al compilar el mismo programa, se resumen en la tabla, donde se detallan los ciclos que emplea cada clase de instrucción y el número de instrucciones de cada tipo, según se utilice el set de instrucciones inicial (ISA1) o el set de instrucciones mejorado (ISA2).

Clase de instrucción	*A*	*B*	*C*
CPI con ISA1 (ciclos)	*2*	*4*	*3*
CPI con ISA2 (ciclos)	*2*	*3*	*4*
Nº instr/programa con ISA1 (miles)	*3*	*3*	*6*
Nº instr/programa con ISA2 (miles)	*3*	*2*	*3*

a) ¿Cuál es el CPI promedio obtenido en cada secuencia de instrucciones (Secuencia con ISA1 y Secuencia con ISA2)?

b) ¿Cuál es la aceleración parcial conseguida en el programa para cada tipo de instrucción?

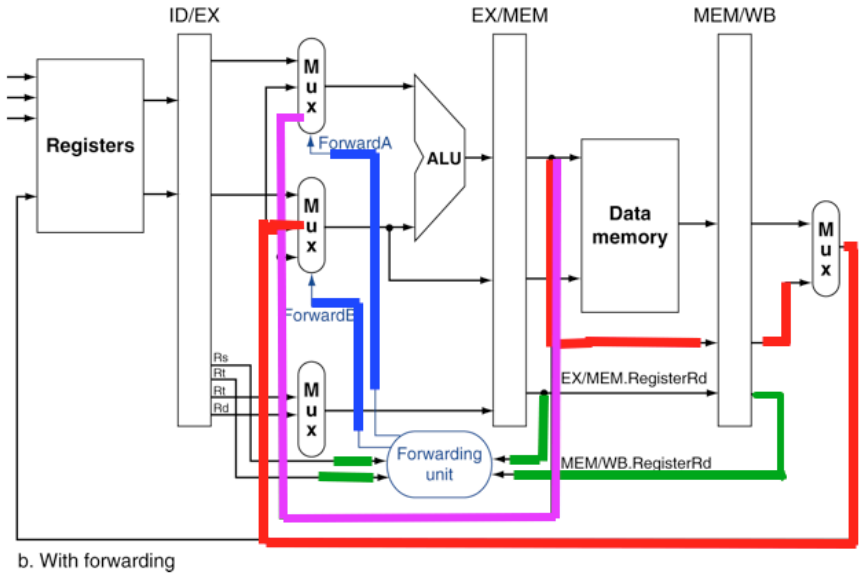
c) ¿*Qué* porcentaje del tiempo de ejecución se aplica cada mejora? y ¿*Qué* porcentaje del tiempo de ejecución no se mejora?

d) ¿Cuál es la aceleración global obtenida al cambiar el set de instrucciones? Justifíquelo utilizando la ley de Amdahl

Hoja en blanco

E.P.S. – U.A.M. – Arquitectura de Ordenadores – 25/Enero/2021

Cuestión 2.- En la unidad de adelantamiento de datos del procesador descrito durante el curso se diseñan caminos de adelantamiento de acuerdo con el esquema de la figura.



Nota: *En la ALU, el operando Rs es el de la entrada superior y Rt el de la entrada inferior. En los multiplexores ForwardA y ForwardB la entrada superior corresponde a la codificación 00, la entrada intermedia a la codificación 01 y la inferior a la codificación 10.*

Sobre la figura se han marcado, con mayor grosor, las líneas que han intervenido para detectar y realizar los adelantamientos necesarios, **durante la ejecución de la instrucción I3** de una secuencia de código.

a) De la secuencia de código se conoce la primera Instrucción (I1 ADD R1,R2,R3). ¿Qué dependencias deben tener I2 e I3 para que en esta secuencia de código se produzca la activación de adelantamientos que se indica en la figura anterior? Complete el cronograma indicando con flechas los adelantamientos que se han realizado.

	1	2	3	4	5	6	7	8
I1: ADD R1, R2, R3	IF	ID	**EX**	**M**	**W**			
I2:								
I3:								

b) Complete el pseudo código que se utilizará en la unidad de adelantamiento, recuadrando los que se correspondan con la activación del adelantamiento de la figura:

Adelantamientos desde la etapa MEM	***Adelantamientos desde la etapa WB***
if (EX/MEM.EscrReg **and** EX/MEM.Reg.Rd ≠ 0 **and** EX/MEM.Reg.Rd = ID/EX.Reg.Rs) **Then** ForwardA = 10 **else** ForwardA = 00 **if** (EX/MEM.EscrReg **and** EX/MEM.Reg.Rd ≠ 0 **and** EX/MEM.Reg.Rd = ID/EX.Reg.Rt) **Then** ForwardB = 10 **else** ForwardB = 00	

c) Indique una secuencia de dos instrucciones con dependencias, en las que sea imprescindible detener la ejecución causando un ciclo de penalización, aunque se activen los adelantamientos disponibles.

Hoja en blanco

E.P.S. – U.A.M. – Arquitectura de Ordenadores – 25/Enero/2021

Apellidos, Nombre	DNI	Test (3p)	C1 (1p)	C2 (1p)	P1 (2.5p)	P2 (2.5p)	TOTAL

Problema 1.- Un procesador MIPS, con arquitectura Harvard y caché unificada para instrucciones y datos, está segmentado en 8 etapas:

F1: Envía la dirección a la caché de instrucciones
F2: Recibe la instrucción desde la caché
D1: Se obtiene la dirección de saltos. *Detección de riesgos.*
D2: Lectura de registros internos. *Calcula la condición en saltos*.
EX: Ejecuta las operaciones aritméticas. Calcula la dirección de instrucciones de memoria.
M1: Envía la dirección a la caché de datos
M2: Lee/Escribe el dato en la caché
WR: Escribe el resultado en el registro interno que corresponda

Se quiere ejecutar el siguiente programa:

```
for (i=1; i<=1001; i++) {
  if (i%7 != 0)
    nm ++;
};
```

```
Dirección           Instrucción
1FC                 addi r1, r0, 1      ; r1 = 1 decimal (índice i)
200                 xor r2, r2, r2      ; r2 = 0 (var nm)
204         L1:     mod r3, r1, 7       ; r3 = r1 mod 7
208                 beq r3, r0, L2      ; salta si 0
20C                 add r2, r2, 1       ; r2 = r2 + 1
210         L2:     add r1, r1, 1       ; r1 = r1 + 1
214                 addi r3, r0, 1001   ; r3 = 1001 decimal
218                 bls r1, r3, L1      ; salta si r1 <= r3
```

Nota: La instrucción "mod" es inventada. Es una instrucción tipo I. Aplica el módulo del valor pasado como inmediato al contenido del registro rs, y guarda el resultado en rt.

a) Identifique los riesgos de datos para esta arquitectura.

b) Se quiere conocer el comportamiento del procesador a la hora de ejecutar instrucciones de salto condicional. Para ello se consideran diferentes estrategias. Completar los siguientes diagramas de ejecución, indicando el número de ciclos perdidos en cada caso. **El beq es efectivo**.

- El pipeline se para cuando se detecta el riesgo.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
BEQ														
NEXT														
NEXT+1														
NEXT+2														
TARGET														

- Predicción no efectiva.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
BEQ														
NEXT														
NEXT+1														
NEXT+2														
TARGET														

- Predicción efectiva.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
BEQ														
NEXT														
NEXT+1														
NEXT+2														
TARGET														

- Se usa un BTB previamente cargado y la predicción del BTB es correcta.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
BEQ														
NEXT														
NEXT+1														
NEXT+2														
TARGET														

c) Suponiendo que instrucciones y datos se encuentran en caché, completar el siguiente diagrama de ejecución sabiendo que es posible leer y escribir un mismo registro en un ciclo de reloj, y que el procesador dispone de unidad de adelantamiento de datos. Respecto a los saltos condicionales, el pipeline se para cuando se detecta el riesgo. **El beq es no efectivo. Indique los adelantamientos con flechas.**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
ADDI																		
XOR																		
MOD																		
BEQ																		
ADD																		
ADD																		
ADDI																		
BLS																		

d) El procesador forma parte de un sistema con memoria total de 4Kbytes, y la caché es de correspondencia directa con 16 entradas y bloques de 16 bytes, y se le ha incorporado una unidad de predicción de saltos estática efectiva.

Indique los campos en los que se divide la dirección para el acceso a la caché en el siguiente recuadro:

Se vuelve a ejecutar el mismo programa (recuerda que el **beq es no efectivo**). Detalle los accesos a memoria que se realizan y enumere los fallos y aciertos producidos completando la tabla dada. La caché está inicialmente vacía.

Nota: Complete una sola iteración del código

OPERACION (instrucción/dato)	DIRECCION	ETIQUETA	INDICE	BYTE/BLOQUE	Fallo/Acierto

Hoja en blanco

E.P.S. – U.A.M. – Arquitectura de Ordenadores – 25/Enero/2021

Apellidos, Nombre	DNI	Test (3p)	C1 (1p)	C2 (1p)	P1 (2.5p)	P2 (2.5p)	TOTAL

Problema 2.- El ARM Cortex-A72 es una microarquitectura que implementa el conjunto de instrucciones ARMv8-A de 64 bits diseñado por ARM y presente en diferentes procesadores comerciales (HiSilicon Kirin 95x, MediaTek Helio X2x, Qualcomm Snapdragon 65x, Rockchip RK3399, NXP i.MX8, Xilinx Versal , etc.). Se puede encontrar en clusters desde uno hasta cuatro cores. Se encuentra esta arquitectura en procesadores comerciales tipo "Big / Little" con otros procesadores ARM más pequeño, como A53. El Cortex-A72 es un pipeline superescalar con ejecución fuera de orden de 3 vías

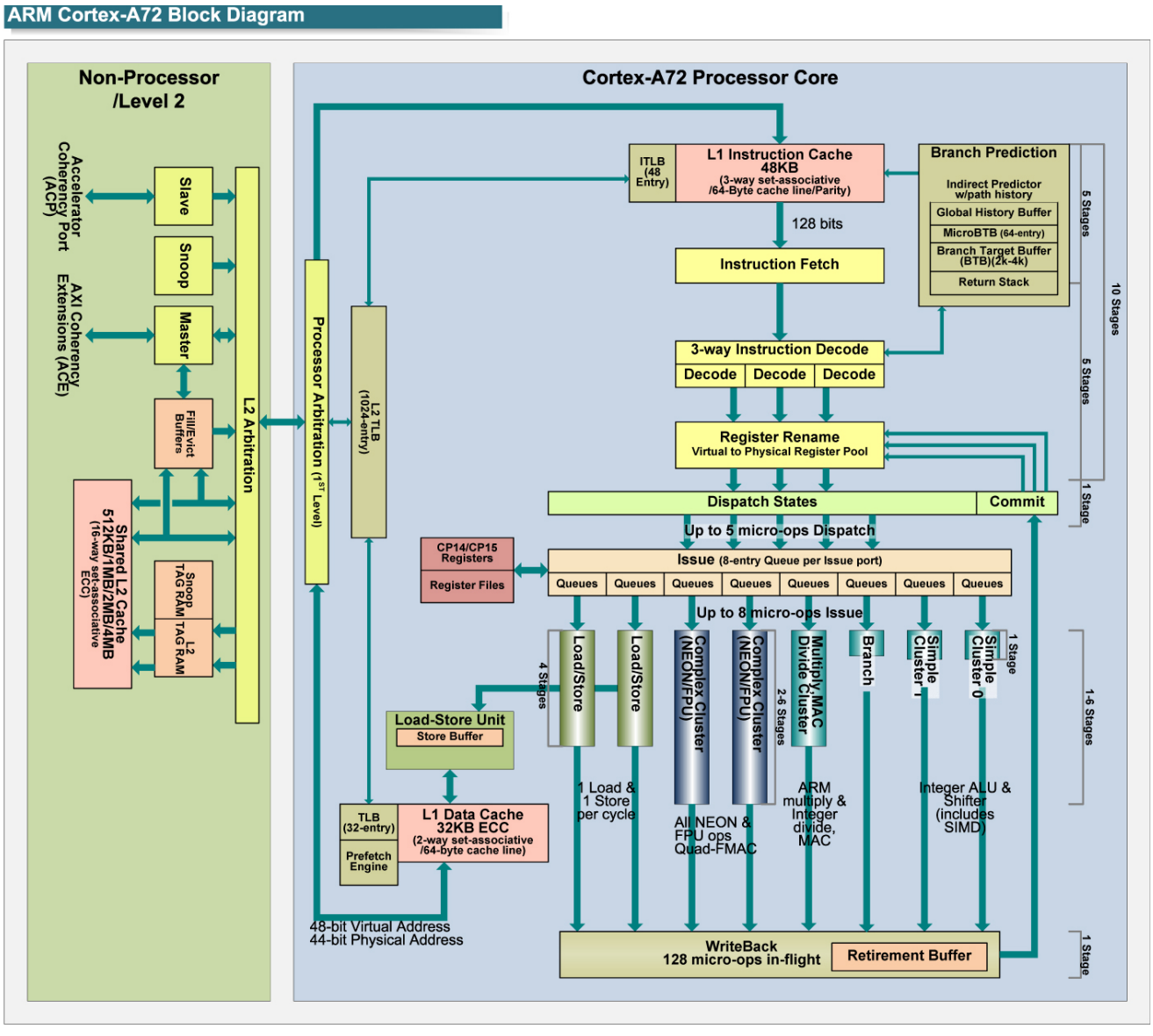


Fig 1. Esquemático del procesador ARM Cortex A.72

El ARM Cortex A72 es un diseño segmentado (pipeline) en más de 15 etapas con ejecución fuera de orden (OoO - *Out of Order*). La etapa de *fetching* captura 128 bits y decodifica 3 instrucciones por ciclo. Estas 3 instrucciones se decodifican en microoperaciones (µops) que luego se introducen en las etapas de *rename/dispatch* (renombramiento/despacho). La unidad de *dispatch* (despacho) puede expedir hasta 5 µops por ciclo que alimentan hasta 8 colas de emisión (IsQ) que a su vez programan µops en 8 pipelines de ejecución. Según ARM, la mayoría de las instrucciones ARMv8 se convierten en una única microoperación (promedio: 1.08 microoperaciones por instrucción).

La ruta de datos, tiene dos ALU simples capaces de operaciones básicas como sumas y desplazamientos (adding and shifting). Las operaciones de multiplicación, división y acumulación de números enteros se gestionan mediante una canalización (pipeline) de enteros multiciclo dedicada (Multi cycle M). Las operaciones de coma flotante, así como SIMD (Single Instruction Multiple Data) y NEON son manejadas por dos pipelines (NEON/FP F0 y NEON/FP F1). Posee una única unidad de saltos (Branch B) y dos AGU (Address Generation Unit - Unidad de Generación de Direcciones) dedicadas. Una para instrucciones de carga (LOAD) y otra de almacenamiento (STORE).
Las IsQ (x) son las siglas de Instruction Issue Queue. (Cola de Instrucciones Emitidas). El número indica la cantidad máxima de instrucciones en la cola.

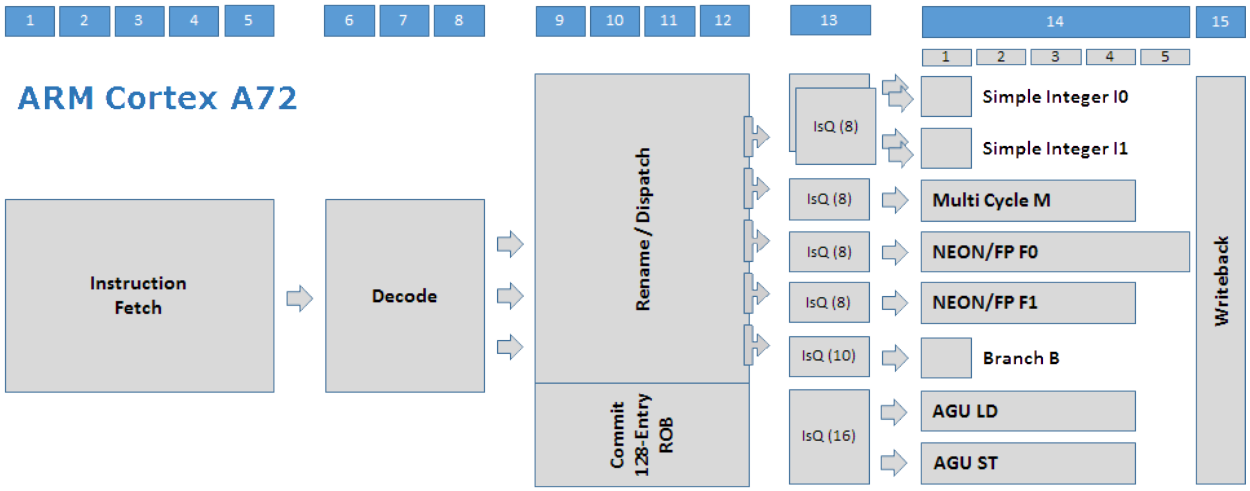


Fig 2. Processor pipeline Schematics

Otras características del procesador:

- Las cachés L1 son 32 KB para datos (asociativas por conjunto de 2 vías) y de 48 KB para instrucción (asociativas de 3 vías) por core. Tamaño del bloque (longitud de línea) de 64 bytes.
- Controlador de caché integrado de nivel 2 de baja latencia (conjunto asociativo de 16 vías), tamaño configurable de 512 KB a 4 MB por clúster. Longitud de línea de 64 bytes.
- Ambas cachés son PIPT (Physically Indexed, Physically Tagged Cach – indexado y etiquetado con direcciones físicas) y utilizan reemplazo LRU
- TLB de primer nivel para instrucciones completamente asociativo de 48 entradas (iTLB) con soporte nativo para tamaños de página de 4 KB, 64 KB y 1 MB.
- El TLB de datos L1 totalmente asociativo de 32 entradas (dTLB) con soporte para tamaños de página de 4 KB, 64 KB y 1 MB
- TLB de segundo nivel unificado asociativos de 4 vías de 1024 entradas por núcleo, admite “hit-under-miss” (permite realizar una búsqueda mientras un fallo se resuelve buscando en las tablas de página).
- Unidad de predicción saltos (branch prediction) que aumenta significativamente el rendimiento. Predicción errónea (Branch Miss) penaliza 15 ciclos

La MMU controla el hardware para el acceso a las tablas de traducción en la memoria. La MMU trabaja con el sistema de L1 y L2 de TLBs para traducir una dirección virtual (VA - 48 bits) a una dirección física (PA). El tamaño máximo de dirección física admitido es de 44 bits usando el modo AArch64.

E.P.S. – U.A.M. – Arquitectura de Ordenadores – 25/Enero/2021

Respecto a esta arquitectura:

1.a. ¿cuántas instrucciones tipo *Load* y cuántas tipo *Store* puede ejecutar por ciclo de reloj? Justifique brevemente.

1.b. ¿Cuál sería el CPI (Ciclos por instrucción) de un programa que solo usa instrucciones de enteros? Considere que no hay ninguna detención por dependencia de datos.

1.c. Si el procesador se configura con páginas de 4KB. ¿qué tamaño de la memoria de instrucciones se puede direccionar sin fallo de iTLB? ¿y con el dTLB? ¿Y el TLB unificado de 2do nivel?

1.d. Se sabe que existen 128 registros de renombramiento. ¿serán suficientes? ¿es posible que existan más de 128 instrucciones emitidas (Dispatched) y no terminadas (Committed)? Justifique su respuesta

E.P.S. – U.A.M. – Arquitectura de Ordenadores – 25/Enero/2021

Rendimiento del sistema:

2.a ¿Cuál es el IPC (instrucciones por ciclo) máximo posible para ese procesador en ausencia de ningún tipo de Riesgo? Justifique brevemente.

2.b. De cara a mitigar la perdida de rendimiento por riesgos de control, ¿Qué estructura agrega este procesador?

2.c. Una predicción de salto fallida, ¿hasta cuantas instrucciones puede afectar en el peor caso? Es decir, ¿cuántas instrucciones podrían estar en el pipeline y deberían ser eliminadas?

2.d ¿Como reduce o elimina los riesgos WAW y WAR este procesador?

E.P.S. – U.A.M. – Arquitectura de Ordenadores – 25/Enero/2021

Respecto de las caches:

3.a. Respecto de la caché L1 de instrucciones y de datos. ¿Cuántos bit se utilizan de índice?

3.b. Sabiendo que las caches de nivel 1 son reales (PIPT – Physically indexed Physical Tagged). ¿Se podría acceder en paralelo la caché y TLB? Justifique brevemente.

3.c. Sabiendo que las caches de nivel 2 es real (PIPT – Physically indexed Physical Tagged). Como se divide la dirección real para el acceso a la cache L2 si esta es de 4 MB. ¿Cuantos comparadores y de cuantos bits son necesarios?

Hoja en blanco