2- Reservation Stations:

* Qi,Qj: Son los punteros a las estaciones de reserva que procesan los operandos para la op. Si están vacios, los operandos ya están listos.
* Vi,Vj : los valores de los operandos
* Busy: Si la RS (reservation Station) está ocupada
* Op: La operación.

Load Buffer:

* Busy: Si está ocupada
* Addr: La dirección para el acceso a memoria (o puntero, si todavía no está disponible)

Store Buffer:

* Busy
* Addr: La dirección a la que se guarda el valor
* Vi: El valor a guardar
* Qi: La RS que producirá el operando con el dato a guardar, si es 0, está listo.

FP Registers:

* Qi: La RS que producirá el resultado para el registro dado. Si está en blanco, es el valor del registro lo que hay que usar.

15-

Sabiendo que:

* SISD son los superescalares que tienen un único flujo de datos, o sea un único proceso. Todo mono núcleo es SISD.
* SIMD son los Illiac IV/SSE I/II/III, inicialmente como procesamiento vectorial y luego para FP y multimedia. Unico flujo de instrucciones, multiples flujos de datos.
* MISD son raros, porqjue es casi imposible explotar múltiple flujo de instrucciones sobre un simple flujo de datos. Un ejemplo sería un sistema tolerante a fallas (aeronavegación)
* MIMD son los multicore o multinucleos. Si están dentro del mismo chasis y conectados por un bus, compartiendo memoria y entrada salida, son fuertemente acoplados. Si Son distribuidos, en red, débilmente acoplados.

Entonces respondo:

1. MIMD fuerte
2. MIMD débil
3. SIMD. Extensión de MMX para FP y int 128-bit
4. SISD. NO CONFUNDIR! Es MT pero mononucleo.
5. SISD. El Mips superescalar tiene único flujo de datos y de instr.
6. MIMD fuerte
7. MIMD débilmente acoplado. Un hipercubo es una topología de red. Son procesadores que tienen su propia jerarquía de memoria y E/s, pero pueden compartir un file register por red.

10)

1. Como ventaja es que las búsquedas pueden resultar mucho más rpadias, dado que no se debe hacer la traducción en pricnipio a dirección física, con la VA se hace todo. Sin embargo, hay problemas de aliasing, donde muchas VA pueden referirse a la misma PA, causando que se cacheen separadamente y generen problemas de consistencia. Otro problema son los homónimos, donde la misma VA mapea a diferentes PA requiriendo del uso de, por ejemplo, un tag con el ID del address space (ASID). Otro problema es que los mapeos virtual físicos tienden a cambiar, lo que requiere un costoso flush.
2. Es la combinación más robusta, dado que aprovecha varias cosas. Por un lado, permite hacer la traducción en simultáneo de la VPN, junto con la búsqueda de la entrada correcta de la cache, ya que el index y offset están contenidos deltro del offset de la VA (o sea, la cantidad de bits de page offset de la Virtual address, tienen que coincidir con la suma de bits de index y offset del caché, sino no tiene mucho sentido..),los cuales no cambian al ser traducidos. Sin embargo, si no queremos problemas, debemos limitar el tamaño de la cache ara que coincida con los bits de offset de la VA. Otra ventaja es que, como se usan PA para los tags, se detectan problemas como homónimos.
3. Tiene a favor su simpleza y e lhecho de que no se tienen problemas de aliasing y homónimos. Sin embargo, es más lenta, debido a que requiere si o si la traducción para realizar la búsqueda.
4. Se considera inútil y no existente, pero el MIPS R6000 lo usa. Son para tecnologías extremandamente rápidas, no aptas para caché grande, por ejemplo la TLB. Debido a la manera de traducción, pueden haber falsos hits, por lo cual es necesario usar tagging para las VA’s.

12)

Es una extensión al set de instrucciones, con instrucciones especializadas para manejar bloques de datos, donde se les debe aplicar una operación.

Esto permite el manejo de audio y video por ejemplo. Un ej: Intel MMX, SSE, Altivec Motorola PowerPC.

Se clasifica en la categoría SIMD, dado que se aplica un único flujo de instrucciones a un gran flujo de datos.

Se implementa con el agregado de registros de un ancho de bits superior (192, 128) al normal, para el manejo de varios datos en simultáneo.

MIPS, por ej, implementa el MDMX, que usa un registro de 192 bit de ancho, el cual actúa como un acumulador par aalgunas operaciones.

13)

A saber:

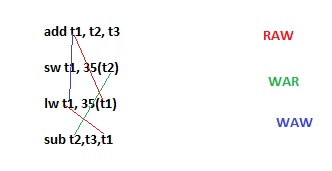
RAW: Son los reads que se hacen a un registro luego de una escritura al mismo.

WAW: Son dos writes consecutivos al mismo registro

WAR: Es una escritura luego de una lectura a un mismo registro.

NOTA: Consecutivo no quiere decir que vengan una después de la otra, sino que pueden estar “cerca”, ponele, 2 3 4 instr desp. (dependen de las etapas del pipeline)

Primero marco los hazards:



a) No tengo ningún hardwar de forwarding ni nada.

(1) IF ID EX MEM

(2) IF ID Stall Stall EX MEM WB

(3) IF Stall Stall ID Stall Stall EX MEM WB

(4) IF ID Stall Stall EX MEM WB

b) Todo hw de forwarding quiere decir que, ni bien calculás algo (por ejemplo, en un add, en la etapa EX ya calculás el res, osea que al final de esa etapa ya lo sabés, y no hace falta esperar al wb para que lo tengas en el registro)

(1) IF ID EX MEM WB

(2) IF ID EX MEM WB

(3) IF ID EX MEM WB

(4) IF ID EX MEM WB

c) En esta en realidad te queda el pipe como B, lo que podrías poner es que los WAW y WAR son riesgos reales solo al momento de utilizar un pipeline múltiple, o sea, con paralelización, dado que si es de un solo flujo como el que pusimos de ejemplo, no pasa nada. En todo caso, te soluciona estos riesgos, peor a fines prácticos, el diagrama sigue siendo el mismo de b).

1) Branch se resuelve en E3

Jump se resuelve en E2

Acá ya sé el res del branch

Pipe branch tomado:

(b) E1 E2 E3 E4

Se pierden dos ciclos por br tomado

(i+1) E1 E2 nop

(i+2) E2 nop

(br tgt) E1

Jump:

(j) E1 E2 E3 E4

Se pierden un ciclo por jump

(i+1) E1 nop nop

(j tgt) E1 E2

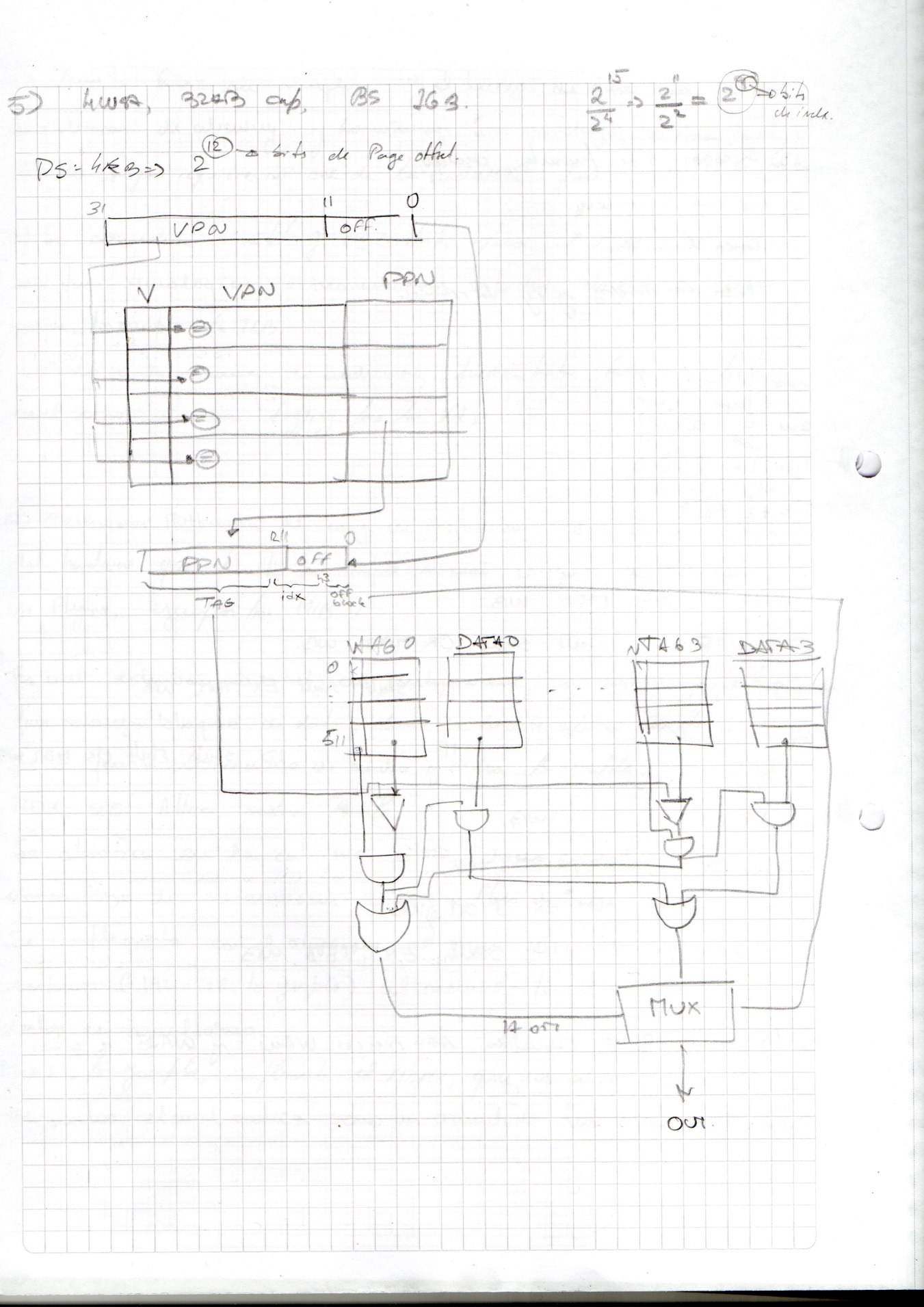
Supongo que en 100 instrucciones tengo 100 clocks sin riesgos.

60% de 25% es 15% tomados 🡪 estos pierden dos ciclos

3% de saltos, pierden un ciclo 🡪

T­100 = 100 + 15x2 + 3x1 => 133

Es un 33% más rápida sin riesgos de control.



2)

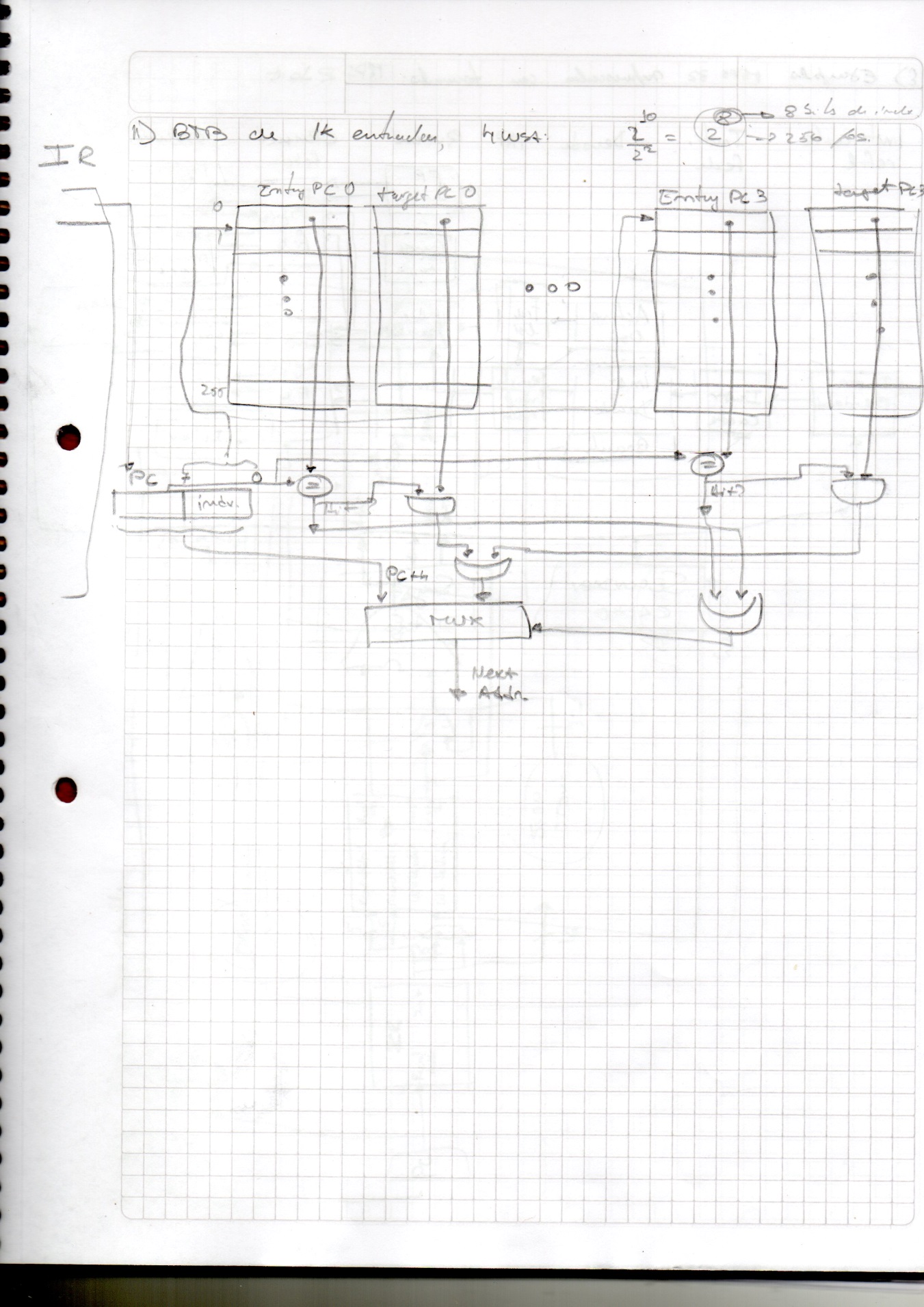
ILP: Se utiliza desde el 1985 el pipelining para solapar la ejecución de las instrucciones y mejorar la performance. A esto se lo denomina Instruction Level Parallelism, debido a que las instrucciones pueden ser evaluadas en paralelo. Es la ejecución de varias instrucciones en paralaelo. Hay muchas técnicas para esto, por ejemplo, loop unrolling

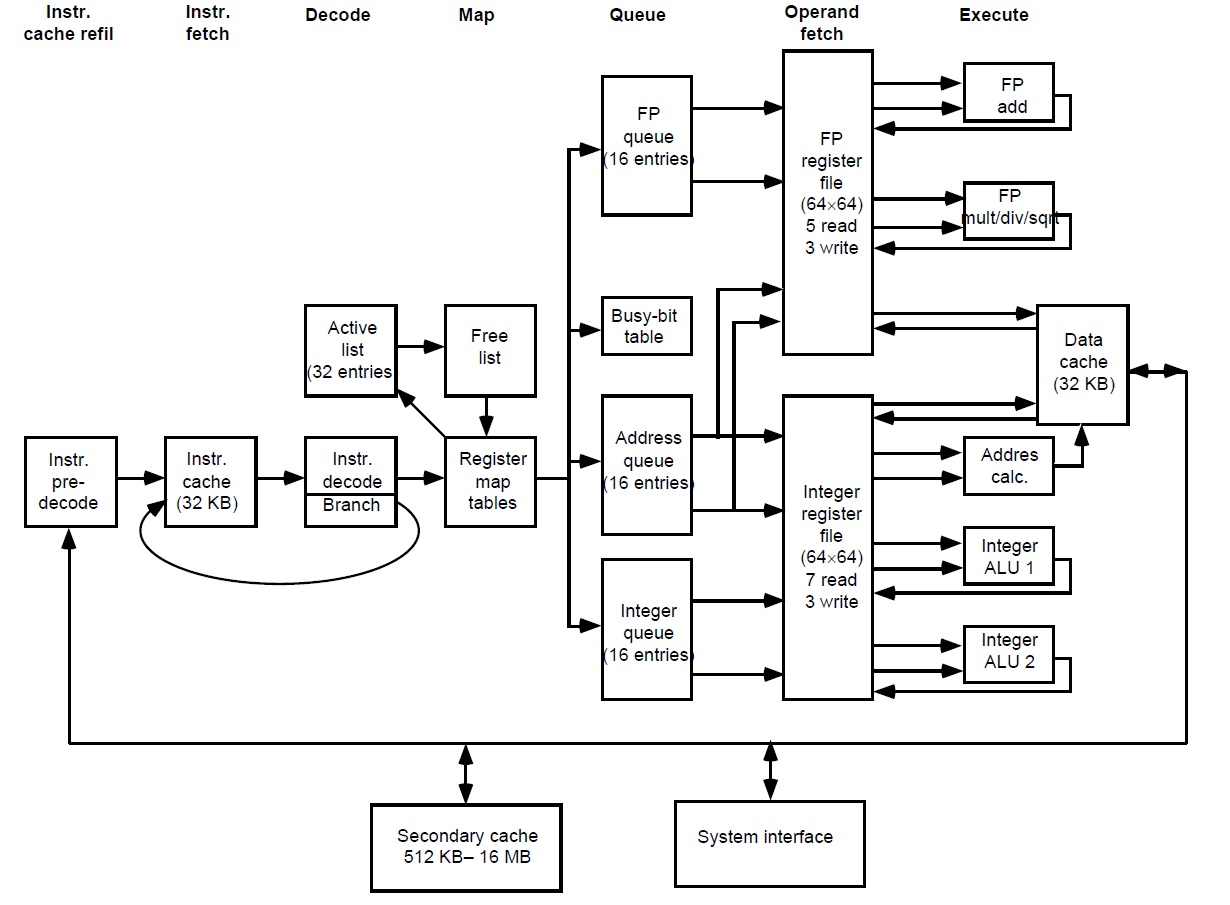
VLIW: El compilador empaqueta en una instrucción larga, a varias instrucciones de la arquitectura del procesador que puedan ejecutarse en paralelo. El compilador sabe la relación entre las instrucciones y las agrupa de forma que se puedan ejecutar en paralelo.

Superescalar: Son procesadores que pueden ejecutar más de una instrucción por ciclo. Se tienen varias líneas de montaje en paralelo donde se poseen etapas que en la práctica se podían hacer en paralelo. Las instrucciones se encolan, y luego se mandan todas las que se puedan en paralelo a las unidades funcionales. Se sigue teniendo un único PC, y un único thread.

En este caso la capacidad de poder ejecutar ciertas instr en paralelo es verificada por el hardware.

Out of order execution: Es el enfoque que utilizan los procesadores de alto desempeño. Las instrucciones se mandan en un orden, pero no necesariamente se completan en ese orden. Las mismas se van completando a medida que tienen la información disponible. Este enfoque permite aprovechar más eficientemente los ciclos. Es lo que pasa, por ejemplo, en el algoritmo de Tomasulo.



5) Ejemplo MIPS32 superescalar con tomasulo: MIPS R10K

El 4) te lo debo, pero por lo que vi, la idea sería aplicar la fórmula de tiempo de acceso promedio a memoria, y después decir que Speedup = Tviejo/Tnuevo.

Saludos EMA.

