Arquitectura de Computadoras. Soución segundo parcial
11 de Abril de 2018

1. Implementa en RTL la instrucción $R_3 \leftarrow M[\text{dir} + A_0]$. Su código de operación es 07. "dir" es una dirección de memoria, su valor se especifica en la siguiente palabra del código de instrucción (la primer palabra tiene el código de operación; la siguiente, el valor de dir). A_0 es un registro de direcciones, su valor se suma a "dir" para obtener el dato que se almacena en R_3.

$$
\begin{aligned}
q_7t_4 &: MAR \leftarrow PC \\
q_7t_4 &: MBR \leftarrow M[MAR], PC \leftarrow pc + 1 \\
q_7t_4 &: MAR \leftarrow MBR + A_0 \\
q_7t_4 &: MBR \leftarrow M[MAR] \\
q_7t_4 &: R_3 \leftarrow MBR, T \leftarrow 0
\end{aligned}
$$

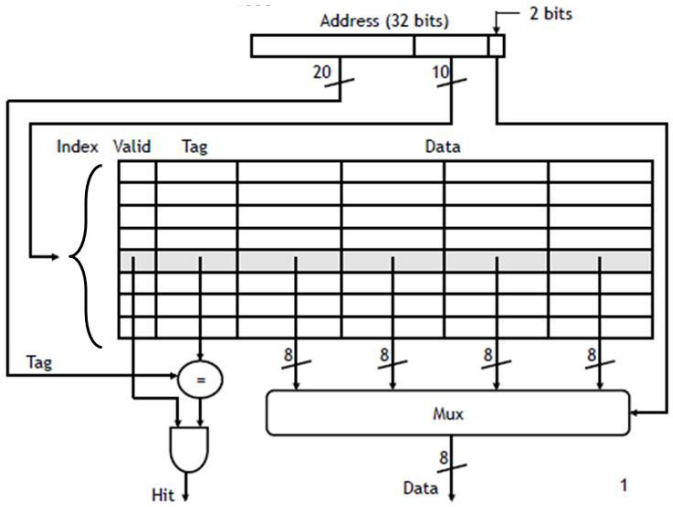
2. Para la memoria cache de la siguiente figura, responde:

   - ¿Cuántos sets (renglones) tiene? $2^{10} = 1024$
   - ¿Cuántos "ways" (bancos) tiene? *Solo un. Es mapeo directo*
   - ¿De qué tamaño es un bloque?c$2^2 = 4$
   - ¿De qué tamaño es la memoria (incluye bits de Tag y valid bit)

     $$1024 \times 4 \times 8 + 21 \times 1024 = 1024(32 + 21) = 54,272 bits$$

   - Si la escritura es *write through*, ¿tiene sentido que haya valid bit?

     Sí. No tiene nada que ver la estrategia de escritura con el hecho de que en caché esté o no el dato (para lectura o escritura) que se necesita.



3. Calcule los ciclos por instrucción (CPI) de un procesador que tiene un CPI promedio para operaciones en ALU de 1.1, para saltos de 3.0 y un tasa de aciertos (hit) en cache de 60%.

   Un hit en cache toma un ciclo, mientras que un fallo (miss) tiene una penalización de 120 ciclos (no hay cache multinivel). Suponga que el 22% de las instrucciones son cargas, 12% almacenamientos (guarda), 20% saltos, y las demás, operaciones en la ALU.

$$
\begin{aligned}
CPI &= CPI_{ALU} \times P_{ALU} + CPI_{JMP} \times P_{JMP} + CPI_{LD/ST} \times P_{LD/ST} \\
&= (1 - (0.22 + .12 + .2)) \times 1.1 + 0.2 \times 3 + (0.22 + 0.12) \times (0.6 + 0.4 \times 120) \\
&= 0.506 + 0.6 + 16.524 \\
&= 17.63
\end{aligned}
$$

4. Considere el siguiente segmento de código, que se ejecuta en un procesador con pipeline de cinco etapas:

   ADD R3, R2, R1
   LD R4, 4(R3)

   (a) Muestre un diagrama de tiempos de la ejecución del pipeline si no se tienen atajos (bloquee el pipeline en caso de conflictos de datos).

   ADD R3, R2, R1: F D X M W
   LD R4, 4(R3): F D D D X M W

   (b) Muestre un diagrama de tiempos de la ejecución del pipeline con atajos.

   ADD R3, R2, R1: F D X M W
   LD R4, 4(R3): F D D X M W
   (No se puede tomar operando (etapa D) sino hasta después del resultado del ADD)

5. Identifique dos maneras en las que *loop unrolling* puede incrementar el desempeño de un programa, y al menos una en la que puede decrementarlo.

   Incremento
   - *Más operaciones disponibles para reordenar código, permite eliminar bloqueos en el pipeline*
   - *Menos saltos reduce desperdicio cuando la dirección del salto no se predice correctamente*

   Decremento
   - *Código más grande, más ciclos fetch*
   - *Si el código es demasiado grande, pueden agotarse los registros disponibles para datos o el código puede no caber en una sola página virtual*

6. Un procesador con despacho dinámico tiene tres unidades funcionales (FU): una LD/STORE que toma dos ciclos, una ADD/SUB, de un ciclo de ejecución, y una MUL/DIV con 2 y 4 ciclos de ejecución, respectivamente. Tiene un solo arreglo de registros y una estación de reservación con un espacio por FU. El procesador tiene una política de emisión y completado de instrucciones fuera de orden (OOI, OOC).

   Comenzando con la secuencia de instrucciones siguiente en el buffer de instrucciones, y con los espacios libres en las estaciones de reservación, identifique el ciclo en el que la instrucción será emitida y en el que escribirá su resultado.

   ```
   LD    R6, 34(R12)
   LD    R2, 45(R13)
   MUL   R0, R2, R4
   SUB   R8, R2, R6
   DIV   R10, R0, R6
   ADD   R6, R8, R2
   ```

   En el diagrama siguiente, se muestra en rojo el momento de la emisión (E_i) y de la escritura (w) para cada instrucción.

			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
LD R6, 34(R12)	F	D	E1	E1	M	W												
LD R2, 45(R13)	F	D	d	d	E1	E1	M	W										
MUL R0,R2,R4	F	D	D	d	d	d	d	d	E3	E3	M	W						
SUB R8,R2,R6	F	D	D	d	d	d	d	d	E2	M	W							
DIV R10,R0,R6	F	D	d	d	d	d	d	d	d	d	d	d	E3	E3	E3	E3	M	W
ADD R6,R8,R2	F	D	d	d	d	d	d	d	d	d	D	E2	M	W				

7. Conteste cierto o falso

C En una arquitectura VLIW, $CPI < 1$
F En una arquitectura Superescalar, $CPI > 1$
F En una arquitectura Vectorial, $CPI < 1$
F En una arquitectura Superpipeline, $CPI \approx 1$
C Las extensiones MMX son una forma de SIMD
F La arquitectura DAXPY es la base de las computadoras Cray
C En general, entre más etapas tenga un procesador pipeline, mejor será su desempeño
F Algunos conflictos de control se resuelven agregando unidades funcionales
F Los conflictos de datos son RAR, RAW, WAR, WAW
C En procesadores digitales de señales (DSP), VLIW es más popular que una Arq. vectorial

8. Seleccione la(s) respuesta(s) correcta(s)

   Se maximiza el paralelismo en una arquitectura superescalar gracias a:

   X El hardware ____ El programador ____ El compilador

   Se maximiza el paralelismo en una arquitectura VLIW gracias a:

   ____ El hardware ____ El programador *X El compilador*

   Se maximiza el paralelismo en una arquitectura EPIC gracias a:

   ____ El hardware ____ El programador *X El compilador*

   Las arquitecturas Harvard ayudan a resolver los conflictos de:

   ____ Datos ____ Control *X Estructurales*

9. Considere el siguiente segmento de código

```
for (i = 0; i < 100; i ++)
   A[i] = ((B[i] * C[i]) + D[i])/2;
```

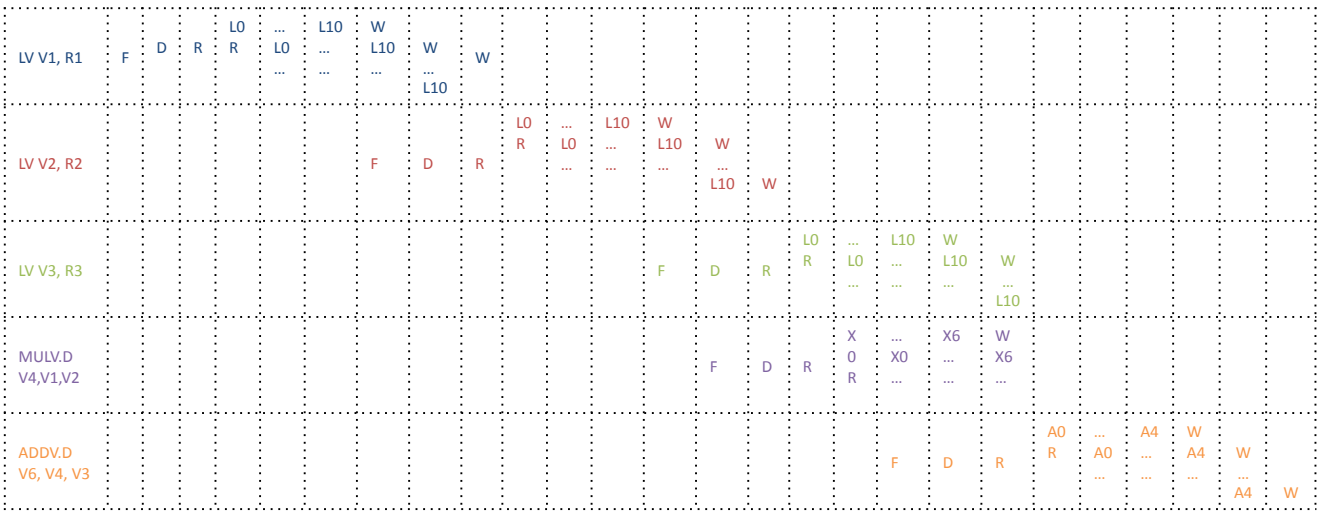
   (a) Escriba un programa en ensamblador que lo ejecute en el menor tiempo posible con las instrucciones VMIPS que se muestran. Suponga que se tienen ocho registros vectoriales con 64 elementos cada uno. Para las operaciones escalares que necesite, utilice los mnemónicos que le sean familiares, pero comente el código

Opcode	No. ciclos	Descripción
MTC1 VLR, R1	1	Carga VLR con el valor de R1
LV Vi,Ri	11,pipeline	Carga Vector i a partir de @R1
SV Ri,Vi	11,pipeline	Guarda Vector i a partir de @R1
MULV.D Vi,Vj,Vk	6,pipeline	Vi <- Vj*Vk
ADDV.D Vi,Vj,Vk	4,pipeline	Vi <- Vj+Vk
SHRVA Vi,Vj,a	1	Vi<-(Vj SHR a)

```
LD R1, #B                ; B = Dir. de memoria donde inicia el arreglo B
LD R2, #C                ; C = Dir. de memoria donde inicia el arreglo C
LD R3, #D                ; D = Dir. de memoria donde inicia el arreglo D
LD R4, #A                ; A = Dir. de memoria donde inicia el arreglo A
MTC1, VLR, #64           ; Procesa máximo número de elementos
LV V1,R1                 ; Carga vectores
LV V2,R2
LV V3,R3
MULV.D V4,V1,V2          ; V4[i] = B[i]*C[i]
ADDV.D V5,V4,V3          ; V5[I] = B[i]*C[i]+D[I]
SHRVA V6,V5,1            ; V6[i] = (B[I]*C[I]+D[I])/2
SV R4,V6
                         ; Stripmining: Ajusta apuntadores y repite
DADDIU R1, R1, #B+64*8   ; Cada elemento ocupa 8 bytes
SLL R2, R2, 9            ; Mismo ajuste, más corto y más rápido
SLL R3, R3, 9
SLL R4, R4, 9
MTC1, VLR, #36           ; Procesa elementos faltantes
LV V1,R1
LV V2,R2
LV V3,R3
MULV.D V4,V1,V2          ; V4[i] = B[i]*C[i]
ADDV.D V5,V4,V3          ; V5[I] = B[i]*C[i]+D[I]
SHRVA V6,V5,1            ; V6[i] = (B[I]*C[I]+A[I])/2
SV R4,V6
```

(b) ¿Cuánto tiempo toma ejecutar el código que desarrolló suponiendo que el procesador tiene memoria traslapada en 16 bancos, no utiliza chaining y sólo tiene un puerto a memora (1 LD o STO por ciclo)

Dado el conflicto estructural, solo se puede hacer una lectura o escritura por ciclo. Además, hay muchos conflictos de datos quye no pueden mejorarse sin chaining. Las únicas instrucciones que pueden traslaparse, son la carga del vector V3 con la multiplicación que le sigue (son UF distintas y no hay conflicto de puertos).
Un segmento del diagrama temporal se muestra en la siguiente figura. El tiempo de ejecución se presenta en la tabla al final de la respuesta.



(c) Repita la pregunta anterior considerando chaining y un solo puerto a memoria

Las instrucciones de carga/guarda no se pueden traslapar pero todas las demás sí. El tiempo de ejecución se presenta en la tabla. Conviene notar que se introduce un retardo forzoso en la instrucción SV hasta que termine la carga del tercer vector.

(d) Repita la pregunta anterior considerando chaining, dos puertos de lectura y uno de escritura

Ahora se pueden ejecutar dos cargas en paralelo. La carga del tercer vector debe retenerse hasta que se libere la UF de acceso a memoria, pero la función SV sí podría ejecutarse en paralelo con una o dos cargas.

Desgraciadamente no se va a poder aplicar esta posibilidad con el ejemplo utilizado pues el VLR tiene distintos valores en la primera y segunda etapa: hubiera sido más conveniente cargar el VLR con un valor de 50 para las dos etapas.

	Inciso (b)		*Inciso (c)*		*Inciso (d)*	
Instr	*Inicio*	*Fin*	*Inicio*	*Fin*	*Inicio*	*Fin*
LD R1, #B	*1*	*5*	*1*	*5*	*1*	*5*
LD R2, #C	*2*	*6*	*2*	*6*	*2*	*6*
LD R3, #D	*3*	*7*	*3*	*7*	*3*	*7*
LD R4, #A	*4*	*8*	*4*	*8*	*4*	*8*
MTC1, VLR, #64	*5*	*9*	*5*	*9*	*5*	*9*
LV V1,R1	*6*	*83*	*6*	*83*	*6*	*83*
LV V2,R2	*81*	*158*	*81*	*158*	*7*	*84*
LV V3,R3	*156*	*233*	*156*	*233*	*82*	*159*
MULV.D V4,V1,V2	*157*	*227*	*92*	*162*	*18*	*88*
ADDV.D V5,V4,V3	*231*	*299*	*167*	*235*	*93*	*161*
SHRVA V6,V5,1	*297*	*362*	*171*	*236*	*97*	*162*
SV R4,V6	*360*	*437*	*234*	*311*	*98*	*175*
Las Op. escalares se traslapan con vectoriales *Solo nuevo valor de VLR debe ser tomado en cuenta*						
MTC1, VLR, #36	*435*	*439*	*309*	*313*	*173*	*177*
LV V1,R1	*436*	*483*	*311*	*358*	*175*	*222*
LV V2,R2	*481*	*528*	*356*	*403*	*176*	*223*
LV V3,R3	*526*	*573*	*401*	*448*	*221 268*	
MULV.D V4,V1,V2	*527*	*569*	*367*	*409*	*187*	*229*
ADDV.D V5,V4,V3	*576*	*611*	*412*	*452*	*232*	*272*
SHRVA V6,V5,1	*609*	*646*	*416*	*453*	*236*	*273*
SV R4,V6	*644*	***691***	*451*	***498***	*237*	***284***

10. (a) ¿Qué significan las operaciones "gather/scatter"?

    Son operaciones que permiten leer/escribir en localidades de memoria no contiguas elementos comunes, típicamente, los elementos de un vector. Se utilizan, por ejemplo, cuando se trabaja con las columnas de una matriz que fue almacenada por renglones, o cuando se utilizan vectores en una memoria traslapada.

    (b) ¿Se puede hacer gather/scatter en una arquitectura vectorial con ISA VMIPS? Explique muy brevemente

    Por supuesto aunque con limitaciones pues la separación . Un ejemplo es el caso de la pregunta 9b, en la que la memoria está dividida en bancos traslapados. Otro ejemplo es la arquitectura Cray 1 vista en clase.

    (c) ¿Se puede hacer gather/scatter en una arquitectura SIMD con CUDA? Explique muy brevemente

    Por supuesto. De hecho, es el caso "natural" pues como el GPU debe verse como una matriz de procesadores, los datos no deben almacenarse en el GPU secuencialmente en un solo bloque (presumiblemente, de un sólo procesador, sino que deben distribuirse en las dimensiones de la matriz de procesadores.

11. Para la siguiente instrucción de CUDA, indique cuántos kernels, cuántos bloques y cuántos hilos se invocan. Explique muy brevemente qué es un kernel, qué es un bloque y qué es un hilo

```
cube<<<25, 64>>>(d_out, d_in);
```

Kernel.- Código a ejecutar en el GPU (típicamente, una función)

Hilo o thread.- Ejecución de un kernel con un índice determinado. El índice permite procesar un subconjunto específico de datos en un arreglo (o matriz)

Bloque.- Grupo de hilos que se ejecuta concurrentemente en un mismo streaming multiprocessor. TIene primitivas básicas de sincronización.

En el ejemplo presentado, hay un kernel (cube) con 25 bloques de 64 hilos cada uno.

12. Una arquitectura GPU permite maximizar:

___ La latencia *X El throughput* ___Ambos: Throughput y latencia ___Ninguno de los dos

13. Las líneas que aparecen en la tabla, son segmentos de código de un programa que calcula el cuadrado de los elementos de un arreglo.

(a) Indique en qué orden deben aparecer las líneas en el código

4	cudaMalloc((void **)&d_in,sz);
2	sz = sizeof(float);
7	cudaFree(d_in);
6	cudaMemcpy(h_out, d_out, sz, cudaMemcpyDeviceToHost);
8	free(h_out);
1	float *h_in,*h_out,*d_in,*d_out;
5	CalculaSq<<< $4, 256$ >>>(d_out,d_in)
3	h_in = (float *)malloc(sz);

(b) ¿Cuántos elementos tiene el arreglo?

Sin información adicional, podemos suponer que tiene $4 \times 256 = 1,024$ *hilos*

(c) Escriba el código de la rutina "CalculaSq", que es la que calcula el cuadrado de los elementos.

(Se omiten instrucciones para detectar condiciones de frontera y control de errores)

```
void cuadrado(float* d_out, float* d_in) {
int idx = blockIdx.x * blockDim.x+threadIdx.x;
float f = d_in[idx];
d_out[idx] = f*f;
}
```

14. ¿Qué estrategia siguen las GPU de NVIDIA si los hilos en un warp divergen en su ejecución?

- Con ayuda del registro de máscara, los hilos se mueven a distintos warps para que no haya divergencia dentro de un mismo warp
- No pasa nada. Es una arquitectura SIMT en la que los hilos pueden divergir

- *Se ejecutan todos los caminos posibles en serie por todos los hilos para que en realidad no haya divergencia*