Control SegPar
20 diciembre de 2005

Considerad un procesador superescalar de grado 4 con las siguientes características:

- Etapas de la segmentación:
  - FDIE…EWC para las arítméticas, siendo el número de ciclos de ejecución (E) dependiente del tipo de instrucción (1 ciclo para las enteras, 2 ciclos para la suma de punto flotante y 4 ciclos para la multiplicación de punto flotante).
  - FDIEMWC para las de memoria, tardando un ciclo de ejecución para el cálculo de la dirección de memoria y 1 ciclo para el acceso a la memoria (que se considerará ideal).
  - FDIEWC para las de salto, para las que supondremos predicción de saltos ideal (es decir, después de la etapa F siempre se sabe la dirección de la siguiente instrucción a ejecutar).
- Unidades funcionales: 1 de aritmética entera, 1 de aritmética en punto flotante (segmentada) y 1 de saltos.
- Puertos de acceso a memoria: 1 puerto con el correspondiente sumador para el cálculo de direcciones. En las instrucciones de `store`, el cálculo de la dirección se realiza tan pronto como se tenga el valor del registro que se utiliza para calcular la dirección, quedando la instrucción almacenada en un buffer de escrituras a memoria en caso de no disponer del dato a escribir. Si un una instrucción de `load` posterior se realiza sobre la misma dirección que un `store` pendiente, el bypass `store-load` del dato se hace en el mismo ciclo que el `store` envia el dato a memoria, eliminando la necesidad de que el `load` haga el acceso.

Dado el siguiente fragmento de código en alto nivel:

```
for (i=1; i<=max; i++) y[i] = a[i] + y[i-1] * x;
```

y su traducción a lenguaje máquina (observaciónes: las líneas en el código anterior indican líneas de cache, los registros $0, $1 y F3 están inicializados con anterioridad):

```
bucle:    ld F1, y-8($0)
          ld F2, a($0)
          mul.f F1, F1, F3; F3 contiene el valor de x
          add.f F1, F1, F2
          st F1, y($0)
          add $0, $0, 8
          sub $1, $1, 1
          jnz bucle
```

Se pide:

a) ¿Cual sería el factor de proporcionalidad que multiplica al número de iteraciones (max) en el tiempo de de ejecución del bucle? Para ello dibujar el cronograma de ejecución

b) Observar que la primera instrucción de `load` es redundante. Si se elimina del código, ¿en cuanto disminuye dicho factor de proporcionalidad?

c) Si el procesador incorporara una nueva instrucción a nivel de lenguaje máquina que realizara de forma combinada la multiplicación y suma en tan sólo 4 ciclos:

```
ADDMUL F1, F3, F2   →  F1 = F1*F3 + F2
```

¿Cual seria el nuevo factor de proporcionalidad que multiplica al número de iteraciones en el cálculo del tiempo de ejecución?