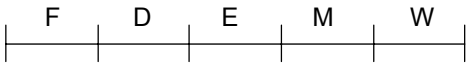
Control de SegPar

14 de noviembre del 2005

Dado un procesador segmentado con las etapas siguientes:



F: búsqueda de instrucciones (*fetch*)
D: decodificación, comprobación de riesgos, lectura de operandos en registros (segunda mitad del ciclo de reloj)
E: operación aritmética, cálculo de la dirección de memoria, o dirección destino de salto y evaluación de condición
M: acceso a memoria
W: escritura de resultado en registro (durante la primera mitad del ciclo de reloj)

El procesador dispone de todos los cortocircuitos (*bypass*) necesarios entre E/E y M/E.

Se pretende evaluar la eficiencia de varios mecanismos de ejecución de las instrucciones de salto condicional, utilizando para ello el código siguiente:

```
j=0;                                  mes:    ld   $2, a($0)
for ( i = 0 ; i < n ; i++) {                  beqz $2, pass
   if (a[i] ≠ 0) then {                       st   $2, b($1)
   b[j] = a[i];                               add  $1, $1, 1
   j++;                               pass:   add  $0, $0, 1
   }                                          sub  $4, $3, $0
}                                             bnez $4, mes
```

Estando inicialmente $0=0, $1=0 y $3=n. Sabemos también que el 40% de los elementos del vector a valen cero.

Se pide calcular el número medio de ciclos por iteración en las situaciones siguientes. Para ello deberéis dibujar el cronograma de ejecución indicando claramente que cortocircuitos se utilizan.

a) Las instrucciones de salto condicional bloquean al procesador durante 2 ciclos (latencia de los saltos de 3 ciclos).
b) Salto con anulación. Al llegar al salto se continúa con la ejecución de las instrucciones siguientes y en caso de saltar, se anulan todas las instrucciones iniciadas.
c) Las instrucciones de salto condicional son con salto retardado (*Delayed Branch*). En este caso, el compilador deberá reordenar el código con el objetivo de optimizar el rendimiento. Muestra cual podría ser el código resultante de aplicar la reordenación.