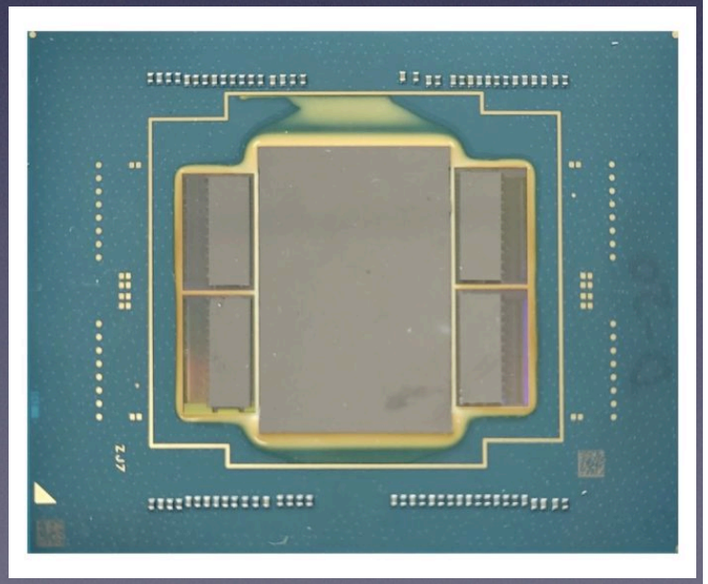
Computación

Noticia del día

Intel Shows 8 Core 528 Thread Processor with Silicon Photonics

Here is an interesting one. Intel has a 66-thread-per-core processor with 8 cores in a socket (528 threads?) The cache apparently is not well used due to the workload. This is a RISC ISA not x86.



Key Technologies

- Core
  - 66 hardware compute threads per core
  - 192KB cache (i$+d$)
  - 4MB scratchpad SRAM
- Socket
  - 8 cores
  - 32 optical I/O ports at 32GB/s/dir
  - 32GB custom DDR5-4400 DRAM
- Sled
  - 16 sockets in an Open Compute Project (OCP) sled form factor,
  - 16TB/s/dir optical bandwidth
  - 0.5 TB DRAM
- Package
  - Multi-die package
  - Co-packaging of electrical to optical dies
- Network
  - HyperX network using high radix, low diameter switches
  - Optical links

Estructuras de control
Sentencia IF()

La forma de hacer preguntas en Fortran
(o cualquier otro lenguaje)

Sentencia IF()

- La idea es que puedo hacer una pregunta y a partir de las distintas respuesta hacer que mi programa tome caminos diferentes.

- Por ejemplo en una ecuación de segundo grado $Ax^2 + Bx + C = 0$, tengo que calcular el discriminante $B^2 - 4AC$ y según si es positivo, negativo o igual cero tendré soluciones muy diferentes (2 soluciones reales, 2 complejas o una sola, en cada caso)

Hay 3 tipos de formas de la sentencia IF() en FORTRAN, sólo veremos las dos que se usan en la actualidad: El IF() sentencia y el IF() en bloque.

IF() sentencia

Se usa “if(algo) sentencia”, se pregunta si lo que está en el paréntesis es verdadero se ejecuta la sentencia. Vamos poner “algo” en el paréntesis que da un resultado lógico (verdadero o falso). Vale entonces el álgebra de Boole.

Para ello si queremos comparar números, tendremos que usar operadores de relación:

$$> , \geq , = , \neq , < , \leq$$

Pero el FORTRAN provenientes de computadores donde esos símbolos no estaban en el teclado, los escribe de esta manera:

Operador	Escritura en Fortran
$>$	.GT.
$\geq$	.GE.
$=$	.EQ.
$\neq$	.NE.
$<$	.LT.
$\leq$	.LE.

Presten atención a los puntos que tienen los operadores antes y después del nombre. Los puntos están para que el compilador no los confunda con variables que tengan esos nombres.

Uso del IF() sentencia

En este caso se pregunta:
IF(algo) “sentencia que se ejecuta”

Ejemplo:

```
IF(A.GT.10) A= B**2 + C**2
```

O también se podría hacer:

```
Logical L
L=A.GT.10
IF(L) A= B**2 + C**2
```

Es decir puedo usar una lógica bastante sofisticada para tomar decisiones.

Sentencia GOTO

```
      GOTO 99
      !estás líneas se saltean
99    sigue el programa
```

Esta sentencia mal usada provoca que los programas sean difíciles de leer, corregir o reutilizar. En muchos libros y artículos suele tener una muy mala prensa

Pero es muy útil para casos como este:

```
    DO I=1,10000
        IF(algo) goto 20
    ENDDO
20 continue
```

Vemos como calcular la serie de Taylor del coseno utilizando esta sentencia.

$$Cos(x) = \sum_{i=1}^{\infty} \frac{(-1)^i x^{2i}}{(2i)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots + \left|error(\xi)\right|$$

Con la idea de indicarle al programa la precisión del resultado, ya que el error se lo podríamos dar como entrada o fijarlo en un valor que nos interese.

Fijamos el error en un ejemplo como error=1E-08

```
      Program cos_de_angulo [1]

      write(*,*) 'ingrese el valor del ángulo n'
      read(*,*) omega
      pi=3.14159265358979
      omega=omega/180*pi
      xmax=pi/2

      coseno=0
      do i=0,1000000

C      Calculo el factorial para el término y para del error: n! y (n+2)!
       facto=1.
       do ii=1,2*i
        facto=facto*ii
       enddo
       facto2=facto*(i+1)*(i+2)

C      Calculo el termino i
       term=omega**(2*i)*(-1)**(i)/facto

C      Calculo el término y el error
       coseno=coseno+term
       eterm= xmax**(2*i+2)/facto2
_____________________________________________
C      Pregunto si llegué al error deseado
       if(eterm.LT.1E-8) goto 99
      enddo

99      write(*,*) i, coseno, cos(omega)

      end
```

- El programa anterior es muy INEFICIENTE y sobrecalcula innecesariamente
—> piensen como mejorarlo

Operadores Lógicos

Operador	Fortran	Operación
Negación	.NOT.	Cambia el valor de la expresión lógica a su opuesto
Conjunción	.AND.	Cierto únicamente si ambas expresiones lógicas son ciertas
Disyunción Inclusiva	.OR.	Cierto si una de las expresiones es cierta
Disyunción exclusiva	.XOR.	Cierto únicamente si una de las expresiones es cierta
Equivalente	.EQV.	Cierto si ambas expresiones tienen el mismo valor
No equivalente	.NEQV.	Cierto si ambas expresiones no tienen el mismo valor

Podríamos escribir entonces:

```
IF(A.GT.10.AND.B.LE.5) X=Z+1

IF(X(3).EQ.4.3.OR.X(4).GT.0) X(5)=X(3)+X(4)
```

Podría hacer:

Logical C1

C1=a.lt.10+4.and.b.gt.33.4.or.75+4.eq.234

if(C1) a=sqrt(x+y**2)

Prioridades de las operaciones

Tipo de Operación	Operador	Asociatividad	Prioridad
Aritmética	** (potencia)	Derecha a izquierda	1
	*, /	Izquierda a derecha	2
	+,-	Izquierda a derecha	3
Relacionales	.GT., .GE., .EQ., .NE., .LT., .LE.	No tienen	4
Lógicos	.NOT.	Derecha a izquierda	5
	.AND.	Izquierda a derecha	6
	.OR.	Izquierda a derecha	7
	.EQV., .NEQV.	Izquierda a derecha	8

Logical LI
LI=a+b**2.ge.22.3.and.f.lt.4.or.a**2*3.5+16.4.eq.10.22