ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORES

3er curso

Práctica 1.Introducción a la computación de alto rendimiento

Departamento de Ingeniería y Tecnología de Computadores

Universidad de Murcia

Introducción

Los procesadores modernos proporcionan una gran capacidad de cómputo gracias a:

- Múltiples cores
- Ejecución fuera de orden (OoO)
- Procesadores superescalares
- Vectorización
- etc

Para conseguir aprovechar toda esa potencia de cómputo es necesario desarrollar software muy eficiente, lo cual puede
llegar a ser realmente complejo.

Objetivos

• Conocer adecuadamente las características básicas de la CPU y el sistema de memoria del ordenador que utilizamos, y saber cómo usar estas características para calcular el rendimiento y el ancho de banda de memoria **pico** de un procesador multinúcleo.

• Comprender la relación entre el código de alto nivel que Escribimos, el compilador utilizado y el código ensamblador, así como los efectos que tienen las *flags* de compilación en el ensamblador generado.

• Conocer el problema de los riesgos de datos y riesgos estructurales, y observar cómo afecta al rendimiento

• Uso de librerías optimizadas para HPC.

Entorno de trabajo

Utilizaremos la CPU de compiladorintel.inf.um.es (i5-6400) como caso de estudio para la explicación de la práctica.

Usaremos una **imagen Docker** que contiene los compiladores que utilizaremos para la práctica:

- gcc 11.0.1
- icc 2021.3.0
- clang 12.0.0

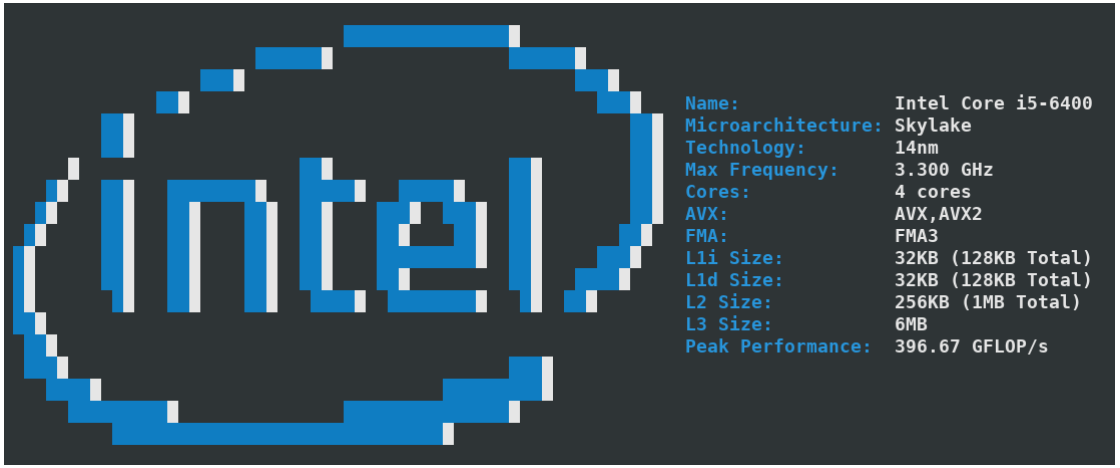
La imagen Docker puede generarse fácilmente con el script proporcionado. También es posible instalar manualmente los compiladores para realizar la práctica (y no usar Docker).

No se recomienda el uso de máquinas virtuales.

Características de la CPU (I)

Existen multitud de herramientas para obtener información del procesador:

- cat /proc/cpuinfo
- lscpu
- cpufetch



cpufetch en compiladorintel.inf.um.es (ejecutado con la opción --accurate-pp)

También es muy interesante consultar la información en ciertas webs:

- ark.intel.com: Información sobre CPUs Intel
- wikichip.org: Información técnica muy detallada sobre microarquitecturas

Características de la CPU (II)

Nombre	`compiladorintel.inf.um.es`	`hostname -f`
Sistema operativo	Ubuntu 14.04.6 LTS	`/etc/os-release`
CPU		
Modelo	Intel® Core™ i5-6400 CPU @ 2.70GHz	`lscpu`, `lshw`
Microarquitectura	Skylake	`ark.intel.com`, `cpufetch`
Número de cores	4	`lscpu`, `cpufetch`
SMT / HyperThreading	No	`ark.intel.com`
Frcuencia mínima	800 MHz	`lscpu`
Frecuencia máxima	3300 MHz	`lscpu`, `cpufetch`
Fecha de lanzamiento	Tercer trimestre de 2015	`ark.intel.com`
Litografía	14 nm	`ark.intel.com`, `cpufetch`
TDP	65 W	`ark.intel.com`
Caché L1 instrucciones	32 KiB (64 conjuntos, 8 vías, 64 bytes/bloque) por core (privada)	`lscpu`, `lshw -C memory`
Caché L1 datos	32 KiB por core (64 conjuntos, 8 vías, 64 bytes/bloque, privada)	`lscpu`, `lshw -C memory`
Caché L2	256 KiB por core (1024 conjuntos, 4 vías, 64 bytes/bloque, privada)	`lscpu`, `lshw -C memory`
Caché L3	6 MiB en total (8192 conjuntos, 12 vías, 64 bytes/bloque, compartida)	`lscpu`, `lshw -C memory`
Extensiones ISA soportadas	SSE4.1/4.2, AVX2 y otras	`ark.intel.com`, `lscpu`
Tamaño de registros vectoriales	256 bits (8 floats ó 4 doubles) `AVX2`	`lscpu`, `cpufetch`
Unidades FPU (vectoriales)	2 Add, 2 Mul	`wikichip.org`
Unidades de FMA	2	`wikichip.org`
Memoria		
Total instalada	8 GiB (1 DIMMs)	`lshw -C memory`
Controladores	1	`lshw -C memory`
Canales/controlador	2 (1 en uso solo)	`ark.intel.com`, `lshw -C memory`
Frecuencia	2133 MHz	`lshw -C memory`

Características detalladas del servidor *compiladorintel.inf.um.es*

Rendimiento pico (I)

$$Rendimiento\_pico_{cores/procesador,procesadores}(tipo) = elementos/instrucción \times operaciones/instrucción \times instrucciones/ciclo \times frecuencia_{cores/procesador} \times cores/procesador \times procesadores$$

- tipo:
  - Indica si usamos simple precisión (SP) o doble precisión (DP)

– elementos/instrucción:
  - Número de operaciones que son ejecutadas simultáneamente por una instrucción (=1 si escalar, si SIMD entonces = ancho UV)

– operaciones/instrucción:
  - Número de operaciones por instrucción (=1, pero si FMA entonces 2)

– instrucciones/ciclo:
  - Número de instrucciones que pueden ser ejecutadas en un ciclo (Igual al n.º de cauces o de UV)

– frequencia:
  - Frecuencia del procesador (usando unidades vectoriales y usando los cores indicados)

– cores/procesador:
  - nº de núcleos físicos que tiene cada socket del procesador.

- procesadores:
  - n.º de sockets que tiene el procesador

Rendimiento pico (II)

$$\begin{aligned} Rendimiento\_pico_{4,1}(\text{sp}) &= 8\ (elementos/instrucción) \times 2\ (operaciones/instrucción) \\ &\quad \times 2\ (instrucciones/ciclo) \times 3\,100\ \text{MHz}\ (frecuencia_4) \\ &\quad \times 4\ (cores/procesador) \times 1\ (procesadores) \\ &= 396{,}800\ \text{GFLOPS} \end{aligned}$$

- tipo:
  - Indica si usamos simple precisión (SP) o doble precisión (DP)
- elementos/instrucción:
  - Número de operaciones que son ejecutadas simultáneamente por una instrucción (=1 si escalar, si SIMD entonces = ancho UV)
- operaciones/instrucción:
  - Número de operaciones por instrucción (=1, pero si FMA entonces 2)
- instrucciones/ciclo:
  - Número de instrucciones que pueden ser ejecutadas en un ciclo (Igual al n.º de cauces o de UV)
- frequencia:
  - Frecuencia del procesador (usando unidades vectoriales y usando los cores indicados)
- cores/procesador:
  - nº de núcleos físicos que tiene cada socket del procesador.
- procesadores:
  - n.º de sockets que tiene el procesador

Ancho de banda pico a memoria (I)

$$Ancho\_banda\_pico = controladores\_memoria \times canales/controlador \times bytes/transferencia \times frecuencia\_memoria$$

- controladores_memoria
  - Número de controladores de memoria
- canales/controlador
  - Número de canales por controlador de memoria
- bytes/transferencia
  - Ancho de banda del canal (en bytes)
- frecuencia_memoria
  - Frecuencia máxima de la memoria

Ancho de banda pico a memoria (II)

$$\begin{aligned} Ancho\_banda\_pico &= 1\ (controladores\_memoria) \times 1\ (canales/controlador) \\ &\quad \times 8\ (bytes/transferencia) \times 2133\ \text{MHz}\ (frecuencia\_memoria) \\ &= 17{,}064\ \text{GB/s} \end{aligned}$$

- controladores_memoria
  - Número de controladores de memoria
- canales/controlador
  - Número de canales por controlador de memoria
- bytes/transferencia
  - Ancho de banda del canal (en bytes)
- frecuencia_memoria
  - Frecuencia máxima de la memoria

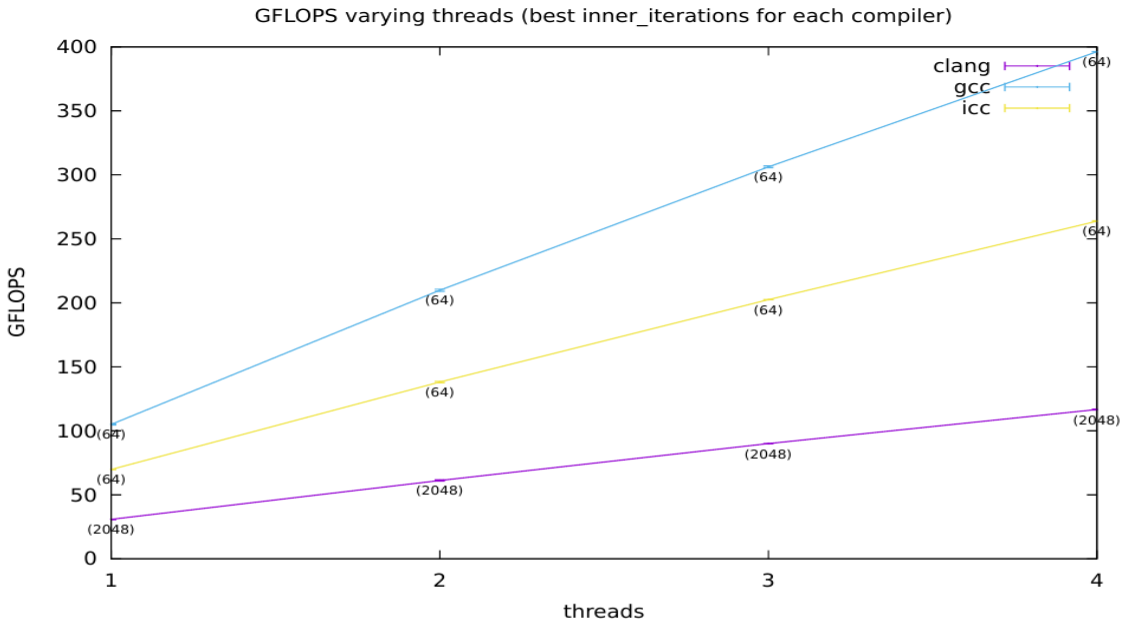
El papel del compilador (I)

"Un programador con experiencia escribiendo una aplicación en lenguaje ensamblador puede superar cualquier código producido por un compilador"

El papel del compilador (II)

- Hoy día los compiladores son capaces de generar código muy eficiente, pero hay que ayudarlos (o al menos no entorpecerlos)
- Opciones de optimización generales:
  - -O1
  - -O2
  - -O3
- Opciones de optimización dependientes de la microarquitectura o conjuntos de instrucciones:
  - - march=XXX y -mtune=XXX (en el caso de gcc y clang)
  - - xhost (en el caso de icc)

Rendimiento pico – Experimental



	Hilos			
Compilador	1	2	3	4
gcc (64)	105.21±0.16	209.73±0.61	306.71±0.09	396.10±0.32
icc (64)	69.59±0.18	135.63±4.01	202.50±0.06	263.93±0.30
clang (2048)	30.92±0.03	61.64±0.12	89.79±0.34	116.89±0.00

Resultados de peak-flops.cpp en *compiladorintel.inf.um.es*

El uso de *intrinsics* evita el problema de obtener distinto rendimiento con diferentes compiladores. Usar la herramienta *peakperf*.

Ancho de banda de memoria pico – Experimental

- Usamos el programa peak-mem.cpp
  - copia manual de arrays en C++
  - la librería *memcpy*

- icc
  - MB (copy) = 13,00±0.02 GB/s
  - MB (memcpy) = 12,98±0,01 GB/s

- gcc y clang
  - MB (copy) = 9,11±0.01 GB/s.
  - MB (memcpy) = 12,84±0,02 GB/s
  - Vemos que en el mejor

- El rendimiento del subsistema de memoria se ve muy afectado por la implementación hardware, el ruido electromagnético en las conexiones, y una variedad de otros factores complejos del sistema.
- Por ello, solo se puede llegar al 75% del valor pico calculado.

Riesgos de datos y riesgos estructurales (I)

- Un problema común que limita el rendimiento obtenido por una aplicación es la existencia de dependencias de datos entre las operaciones (riesgos de datos), o la limitación del número de unidades funcionales (riesgos estructurales)
- Este problema lo vamos a mostrar con el programa *data-hazards.cpp* para descubrirlo de forma experimental:

```
for (size_t j = 0; j < total_iterations; ++j) {
  a1 = a1 + 1.0f;
}
```

data-hazards.cpp cuando num_vars=1

```
for (size_t j = 0; j < total_iterations; ++j) {
  a1 = a1 + 1.0f;
  a2 = a2 + 1.0f;
}
```

data-hazards.cpp cuando num_vars=2

- Además, esta aplicación nos permitirá conocer cuántas unidades funcionales tiene nuestra CPU, y cuál es su latencia

Riesgos de datos y riesgos estructurales (II)

$$T = \frac{total\_iterations \times c}{frecuencia_1}$$

$$0{,}78\ \text{s} = \frac{640000000 \times c}{3300\ \text{MHz}} \Rightarrow c = \frac{0{,}78\ \text{s} \times 3300 \times 10^6\ \text{Hz}}{640000000} = 4{,}02$$

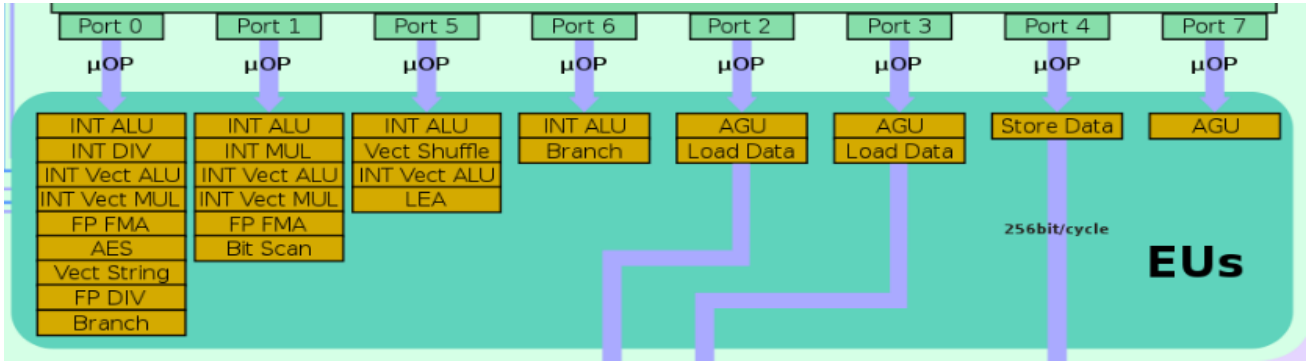
Los resultados indican que:
- Tenemos 2 unidades funcionales de suma en coma flotante de simple precisión.
- Las unidades funcionales tienen una
latencia de 4 ciclos de reloj.

vars	Tiempo (s)	GFLOPS
1	0.78±0.00	0.82±0.00
2	0.78±0.01	1.63±0.02
3	0.78±0.01	2.45±0.02
4	0.78±0.00	3.29±0.00
5	0.78±0.00	4.09±0.03
6	0.78±0.00	4.94±0.00
7	0.78±0.00	5.77±0.00
8	0.78±0.00	6.55±0.04
9	0.87±0.00	6.58±0.00
10	0.97±0.00	6.58±0.00
11	1.07±0.00	6.56±0.02
12	1.17±0.00	6.56±0.02
13	1.27±0.00	6.57±0.01
14	1.36±0.00	6.57±0.01
15	1.47±0.01	6.52±0.03
16	1.75±0.00	5.84±0.01
17	1.77±0.01	6.16±0.02
18	1.76±0.00	6.55±0.01
19	1.91±0.00	6.37±0.01
20	1.96±0.00	6.55±0.02

Resultados de data-hazards.cpp en *compiladorintel.inf.um.es*

Riesgos de datos y riesgos estructurales (III)

También podemos conseguir ambos valores de forma teórica:



https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(client)

Skylake

CVT(T)SD2SI	r32/64,x	2	2	p0 p1	6	1	
CVT(T)SD2SI	r32,m64	3	3	2p01 p23		1	
VCVTPS2PH	x,v,i	2	2	p01 p5	5-7	1	F16C
VCVTPS2PH	m,v,i	3	3	p01 p4 p23		1	F16C
VCVTPH2PS	v,x	2	2	p01 p5	5-7	1	F16C
VCVTPH2PS	v,m	1	2	p01 p23		1	F16C
Arithmetic							
ADDSS/D PS/D							
SUBSS/D PS/D	x,x / v,v,v	1	1	p01	4	0.5	
ADDSS/D PS/D							
SUBSS/D PS/D	x,m / v,v,m	1	2	p01 p23		0.5	
ADDSUBPS/D	x,x / v,v,v	1	1	p01	4	0.5	SSE3
ADDSUBPS/D	x,m / v,v,m	1	2	p01 p23		0.5	SSE3

https://www.agner.org/optimize/instruction_tables.pdf (Pag. 275)

Bibliotecas HPC (I)

- Optimizar aplicaciones para obtener un alto rendimiento puede ser emocionante, pero también es una tarea muy compleja.
- Puesto que en muchos campos de la informática es necesario aprovechar el rendimiento del hardware todo lo posible, ya existen librerías optimizadas para diferentes problemas:
  - BLAS (álgebra lineal)
  - FFTW (transformada discreta de Fourier)
  - NAMD (bioinformática)
  - GROMACS (bioinformática)
- Estas librerías llevan muchos años siendo desarrolladas y mejoradas por especialistas; en líneas generales, su rendimiento es excelente.
- ¡No reinventemos la rueda! Si ya existe una librería optimizada para hacer una cierta tarea, no implementemos nosotros lo mismo desde cero.

Bibliotecas HPC (II)

- Realizamos un experimento con multiplicación de matrices
- Utilizamos la librería BLAS
- Comparamos diferentes versiones contra la de BLAS

	basic	ikj	basic_omp	ikj_omp	blas
750	1.66±0.00	14.76±0.05	6.13±0.18	52.29±0.05	294.02±1.83
1000	1.64±0.00	13.80±0.09	6.13±0.01	50.97±0.10	290.84±1.95
1250	1.36±0.05	12.41±0.06	5.07±0.01	40.98±0.11	304.90±0.56
1500	1.27±0.00	10.71±0.09	4.57±0.48	27.80±0.01	288.12±0.92
1750	1.05±0.00	9.19±0.01	4.03±0.02	25.34±0.12	296.09±0.44
2000	1.34±0.01	8.40±0.01	5.10±0.02	25.52±0.09	304.61±1.93
2250	1.05±0.00	8.00±0.02	3.94±0.11	22.69±1.39	304.70±0.64
2500	1.23±0.00	7.80±0.01	4.71±0.01	25.70±0.31	325.79±0.23
2750	1.09±0.00	7.58±0.01	4.11±0.01	23.30±0.64	311.77±0.45
3000	1.30±0.00	7.35±0.08	4.89±0.12	24.27±0.60	313.43±0.48
4000	1.06±0.00	7.14±0.03	4.21±0.06	19.67±2.36	321.79±0.32
5000	1.24±0.00	6.76±0.02	4.25±0.03	21.70±1.32	341.49±1.71

Resultados de las diferentes implementaciones de multiplicación de matrices en *compilador.intel.inf.um.es*. Los resultados se muestran en GFLOP/s

Ejercicios pedidos

- Elige un ordenador diferente de *compiladorintel.inf.um.es* y contesta a las siguientes preguntas:
  - Detalla las características relevantes del ordenador elegido (microarquitectura de CPU y memoria).
  - Calcula el rendimiento pico teórico para operaciones en coma flotante de simple precisión.
  - Calcula ancho de banda pico que puede alcanzar el sistema de memoria.
  - Mide el rendimiento pico en coma flotante de simple precisión del ordenador elegido utilizando los benchmarks peak-flops.cpp y peakperf.
  - Mide el ancho de banda pico de memoria del ordenador elegido utilizando el benchmark peak-mem.cpp.
  - Calcula la latencia de las sumas en coma flotante de simple precisión y trata de determinar de cuántas unidades funcionales dispone cada núcleo para esta operación.
  - Explica los resultados numéricos obtenidos en la ejecución de data-hazards.cpp (el valor mostrado en cada caso como «result = ...», no el tiempo medido).

Entrega

- Tras finalizar los ejercicios anteriores, se pide entregar lo siguiente:
  - Un documento en formato PDF que será el documento principal utilizado a la hora de evaluar la práctica. Dicho fichero debe incluir:
    - Información completa y fehaciente sobre la autoría de la práctica.
    - Las respuestas a los ejercicios solicitados.
    - Un breve informe comentando los aspectos positivos de la práctica, así como cualquier aspecto negativo y cosas que has echado en falta en la misma. La longitud del informe no debe exceder las 2000 palabras
  - El código fuente de cualquier programa o script desarrollado (o modificado) para la realización de la práctica. Estos ficheros se mirarán opcionalmente para comprobar cualquier aspecto que no esté claro en el documento PDF.
  - Un fichero de texto llamado README identificando todos los ficheros fuente incluidos con instrucciones claras para su compilación.

Criterios de evaluación

- Esta práctica se evaluará teniendo en cuenta los siguientes criterios de evaluación (sobre 10 puntos):
  - Realización y corrección de los apartados pedidos (6 puntos)
  - Presentación y claridad de las explicaciones (1 punto)
  - Aportación de ideas originales en las explicaciones a los apartados realizados (1 punto)
  - Realización de alguna actividad extra relacionada con los ejercicios de la práctica demostrando curiosidad (1 punto)
  - Comentarios sobre la práctica, incluyendo aspectos negativos y positivos (1 punto)

ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORES

3er curso

Práctica 1.Introducción a la computación de alto rendimiento

Departamento de Ingeniería y Tecnología de Computadores

Universidad de Murcia