WEEK 3

ONDERWERPEN

- typen geheugen
- caching
- opslag (ssd, harddisk)
- het vertalen en starten van een programma
- parallelle architecturen (h/w multi-threading, multicore en GPU)

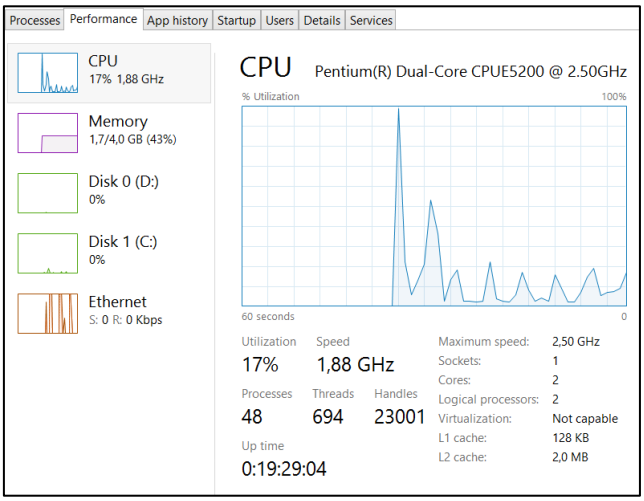
OPDRACHT 1 : VERGELIJKING TYPEN GEHEGEN EN OPSLAG

Een computersysteem heeft diverse mogelijkheden om gegevens op te slaan : DRAM, Flash en (magnetische) schijf of harddisk. Geef voor deze drie typen opslag de access tijd en de kosten per GByte. Geef verder ook de *verhoudingen* in opslagtijd en kosten.

Antwoord
DRAM memory: short access time of 50 to 70 nanoseconds, and cost per GB is $5 to $10.
Flash memory: access times are 100 to 1000 times slower than DRAM, and cost per GB is 7 to 10 times cheaper than DRAM.
Disk memory: access times are 100,000 to 400,000 times slower than DRAM, and cost per GB is 100 times cheaper than DRAM.

OPDRACHT 2 : CPU PERFORMANCE

Via de Windows Task manager kun je mooie plaatjes laten zien m.b.t. de CPU performance :



a) Wat is het verschil tussen 'Cores' en "Logical processors' ?

Antwoord

Stel er zijn 2 cores en 2 logical cores, zoals in de afbeelding. Dit betekent dat er echt fysiek 2 afzonderlijke cores zijn, en elke core heeft 2 hyperthreads. Dus de kernel ziet 4 cores, maar de performance is een stuk minder dan 4 fysieke cores.

b) Tegenwoordig zitten er meestal 2 of 4 cores in dezelfde chip (in hetzelfde doosje). Maar wat betekent dan een CPU belasting van 80% ?

Antwoord

Dit is de gemiddelde belasting (over meerdere logical cores). Dit ook zien via de knop "open resource monitor", dan word de belasting per logical core getoond. (Staat helemaal onderin de window, hier niet zichtbaar)

c) Hoe wordt de CPU belasting gemeten ?

Antwoord

Een beetje lastige vraag. Een CPU is of 'active' of 'idle'. Natuurlijk kan de CPU niet niets doen, maar idle is een speciale taak/proces. Het OS neemt periodiek een sample, en door bij te houden hoeveel de taak idle is geconstateerd kun je berekenen hoeveel % van de tijd de CPU 'active' was. De CPU belasting wordt dus altijd gemeten over een bepaalde periode.

OPDRACHT 3 : CACHING

Cache geheugen is een belangrijk middel om de performance van een computersysteem te verhogen. Gegeven is dat de cache size = 16. De volgend reference string is gegeven, d.w.z een lijst van 8-bit geheugen adressen die achtereenvolgens worden benaderd door de CPU : 3, 180, 43, 2, 191, 88, 180, 14, 181, 43

a) Maak de volgende tabel af. Geef voor elk van deze cache-entries : het binaire adres, de tag en de cache index.

Antwoord

memory address decimal	binary address	tag	index	hit/miss
3	0000 0011	0000	0011	M alleen deze regel gegeven in vraag
180	1011 0100	1011	0100	M
43	0010 1011	0010	1011	M
2	0000 0010	0000	0010	M
191	1011 1111	1011	1111	M
88	0101 1000	0101	1000	M
180	1011 0100	1011	0100	H
14	0000 1110	0000	1110	M
181	1011 0101	1011	0101	M
43	0010 1011	0010	1011	H

Toelichting : 180 is binair 1011 0100. Dus tag is 1011 en index is 0100. Omdat initieel de cache leeg is zijn er in het begin alleen maar misses. Alleen 180 en 43 komen 2x voor, de 2e keer is dus een hit.

b) Beschrijf hoe, na het uitvoeren van de reference string, de inhoud van de cache door onderstaande tabel af te maken :

Alleen de kolom index is gegeven

index	valid-bit (Y/N)	data = value of memory address
0	N	
1	N	
2	V	2

3	V	3
4	V	180
5	N	
6	N	
7	N	
8	V	88
9	N	
10	N	
11	V	43
12	N	
13	N	
14	V	14
15	N	

Toelichting : Alle valid bits komen na het vullen van de cache op Yes. De index-waarden zijn al bij (a) uitgerekend, bijvoorbeeld 180 heeft index = 4.

OPDRACHT 4 : CACHING

Voor een direct-mapped cache ontwerp met 32-bit adressen en een 32-bit database. De onderstaande adresbits worden gebruikt om toegang te krijgen tot de cache. De index is dus a.h.w. gesplist in 2 delen : de index bits 5-9 verwijzen naar een *blok*, en binnen een blok verwijzen bits 0-4 naar een *datawaarde* in dat blok. (In de cache worden dus blokken van aaneengesloten stukken geheugen opgeslagen).

Tag	Index	Offset
31–10	9–5	4–0

a) Hoeveel entries heeft de cache ?

Antwoord: 5 bits voor de index, dus 2^5=32 entries.

b) Hoeveel data kan in de cache worden opgeslagen ?

Antwoord: Een blok heeft 32 entries (bits 0-4) en er zijn 32 entries in de cache (bits 5-9). Dus 2^10 geheugen waarden. Omdat de databus 32 bit is, zal ook een geheugenplaats 32 bit zijn. Oftewel 2^10 * 4 bytes = 2^12 bytes = 4kB.

c) Hoeveel extra (redundante) bits ben je in dit ontwerp kwijt per 32-bit data waarde in de cache ?

Antwoord: Voor elke entry is extra nodig : 1 valid bit en 22 bit voor de tag, is samen 23 bit. Maar een blok bevat 32 waarden. Dus per waarde slechts 23/32 = 0,72 bit.

OPDRACHT 5 : COMPILER EN INTERPRETER

a) Noem drie verschillen tussen een compiler en een interpreter.
Zie de sheets.

b) Beschrijf de stappen die plaatsvinden bij het interpreteren en uitvoeren van een Javascript file door een web browser.

- The script sections of a web page are handled by the browser's JavaScript interpreter (Chrome uses V8).
- When the HTML parser reaches a script element, all that the parser does is read and store the text through the ending </script> tag (or retrieve the file referenced via the src attribute).
- The HTML parser hands the script text off to the JavaScript interpreter.
- The JavaScript interpreter interprets the JavaScript code in the context of the window object, and when done returns to the HTML parser, which can then continue parsing and displaying the page.
- This stop-everything-and-run-the-JavaScript is why some prominent people recommend putting scripts at the bottom of the page to improve the perceived load time.

- It also means that script tags are processed in order, which can be important if one script relies on another.
- If the defer or async attribute is used, script execution can be deferred until later on browsers that support it.
- All scripts on the page are executed within the same global execution context, sharing the same global namespace and memory area (and thus can interact with one another).
- As you can see above, there are two somewhat different situations in which JavaScript code can be run: During the page parsing/rendering process (when a script element that does not use the defer or async attributes is being initially processed), and after the parsing/rendering process (deferred scripts, and code running in response to an event). JavaScript running during the parsing/rendering process can directly output content to the HTML parser via the document.write function. JavaScript running after the parsing/rendering is complete can't do that, of course, but can use the very powerful DOM HTML API to interact with the DOM.

c) Beschrijf de stappen die plaatsvinden bij het interpreteren en uitvoeren van een Java file door de JVM.

- Java runs a virtual machine which is an implementation of a stack engine. It is designed to interpret the bytecodes present in the class file. And when a certain set of bytecodes eg. a loop or a method executes more than a threshold (lets say a million times) Java invokes the compiler which compiles the bytecodes into native code which would run faster.

d) Beschrijf de stappen die plaatsvinden bij het compileren en uitvoeren van een C-file door C-compiler en besturingssysteem.

- Zie http://www.tenouk.com/ModuleW.html
- Preprocessing is the first pass of any C compilation.
- Compilation is the second pass. It takes the output of the preprocessor, and the source code, and generates assembler source code.
- Assembly is the third stage of compilation. It takes the assembly source code and produces an assembly listing with offsets. The assembler output is stored in an object file.
- Linking is the final stage of compilation. It takes one or more object files or libraries as input and combines them to produce a single (usually executable) file.

- Before we can run an executable, firstly we have to load it into memory. This is done by the loader, which is generally part of the operating system.

OPDRACHT 6 : PARALLEL COMPUTING

Tijdens het college zijn een aantal mogelijkheden behandeld om de performance van computers te vergroten :

- caching (in de CPU)
- pipelining
- meerdere instructies tegelijk
- on-chip multithreading
- single chip multi-processors

De ATmage32 datasheet is te vinden op Blackboard. Hiervan hoef je alleen de paragrafen "Features" en "CPU Core" te lezen.

a) Bespreek in hoeverre deze mogelijkheden zijn toegepast in de ATmega32.
Alleen pipelining.

b) Zoek uit welk type CPU er in je PC/laptop zit. Bespreek in hoeverre bovenstaande mogelijkheden zijn toegepast in je machine. (Via device management in Windows kun je het type CPU achterhalen).
Alle bovenstaande, behalve meerdere instructie tegelijk.

OPDRACHT 7 : DE GPU

Bekijk onderstaande Youtube video's :

https://www.youtube.com/watch?v=_cyVDoyI6NE
https://www.youtube.com/watch?v=CdPd5TwM0mI (alleen de eerste 15 min.)

a) Noem twee verschillen tussen een CPU en een GPU.
CPU is geoptimaliseerd voor minimale latency, een GPU voor maximale throughput
GPU bevat veel meer cores (>100), maar GPU cores zijn eenvoudiger (dan CPU cores)
GPU heeft cores hebben meer geheugen (register file)
GPU heeft lagere klokfrequentie
GPU is geoptimaliseerd voor grafische bewerkingen

b) Noem drie toepassingen van de GPU.
Teveel om op te noemen.
https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units#Applications