Today we’ll cover three topics

- Administrivia for 3007 *Computer Systems*: people, class policies

- What is “Systems?”

- Introduction to x86-64 assembly (for your first coursework, handed out today!)
Course Staff and Office Hours

• Instructor:
  – Brad Karp, MPEB 7.20, Mon 6 - 8 PM and Thu 2:30 – 4:30 PM, x30406

• Teaching Assistants:
  – Ahmed Awad, MPEB 7th floor lab, Tue 2 - 4 PM, x TBC
  – Nikola Gvozdiev, MPEB 7th floor lab, Wed 1 - 3 PM, x33670

• Additional office hours:
  – MPEB 7th floor lab, Fri 2 - 4 PM (rotating staff)

• Time reserved for answering your questions
• Outside office hours, please send a private Piazza message to schedule an appointment
Meeting Times and Locations

• Tuesdays 5–6:30 PM
  1-19 Torrington Place, G12
• Thursdays 1–2:30 PM
  Royal National Hotel, Galleon B, all but final week
  IoE, Jeffrey Hall 103 last week

• Lecture will run 90 minutes
• Sometimes lecture will be followed by a 30-minute discussion (e.g., Q & A on a coursework)
• Detailed calendar on class web page
• Reading week: 12th – 16th February, no lecture!
Class Communication

• Class web page
  – http://www.cs.ucl.ac.uk/staff/B.Karp/3007/s2018
  – Detailed calendar, coursework, class policies, announcements/corrections
  – Your responsibility: check page daily!

• 3007 Piazza Page
  – https://piazza.com/ucl.ac.uk/spring2018/computerscience3007/home
  – Important announcements from class staff (also forwarded to you by email)
  – Postings from course staff and students
  – Subscribe using enrollment key
  – You must subscribe (class policy)
  – Your responsibility: check email daily!
Using Piazza

• Please post questions on class material on Piazza, rather than emailing course staff

• Whole class benefits from seeing your question and its answer

• Students are encouraged to answer one another’s questions!

• When discussing something private (e.g., your marks, or details of your specific solution to a coursework), mark your post as private, so only course staff see it!
Readings, Lecture Notes, A Few Research Papers

• Readings must be read before lecture; lectures **assume you have done so**
  – All readings are examinable

• Lecture notes will be posted to the class web site just after lecture

• Class calendar shows readings day by day...
Texts for 3007

• Required text:

• Recommended text:

• Waterstone’s and amazon.co.uk sell both texts
Classic Research Papers

• Will be available on class web page; print these yourselves:
  – LZW Compression (used in CW3)
  – Reflections on Trusting Trust
  – The UNIX Time-Sharing System
  – [likely 4th paper TBA]
How You Will Be Evaluated

• Final mark components:
  – Five programming coursework: 30%
  – In-lecture mid-term exam: 10%
  – Final exam: 60%

• Each coursework is worth 6%

• Midterm, in-class exam:
  – 1.5 hours, 22\textsuperscript{nd} February
  – Absence must be cleared by a doctor’s note (or similar documentation)
Our Late Work Policy

- *N.B.* that our policy differs from that of other CS classes!

- Each student granted **five late days** for the entire term
  - For every day late **or fraction thereof, including weekend days**, 10% of marks deducted
  - Each late day “cancels” one day of lateness
  - Goal: give you flexibility, e.g., in case you can’t find a bug, or encounter unexpected other snag
  - You choose how many late days to use when turning in a coursework late **and state on your coursework**
  - Must use whole late days—cannot use fractional ones!
Our Late Work Policy (cont’d)

• If submission more than two days late after taking late days into account, zero marks

• Programming courseworks turned in online via GitHub; may be submitted 24/7

• **Late days give you flexibility.** No other extensions given on coursework, unless for unforeseeable, documented severely extenuating circumstances!
Academic Honesty

• All courseworks must be completed individually
• May discuss understanding of problem statement and a general sketch of approach
• May not discuss details of solution
• May not show your solution to others (this year or in future years)
• May not look at others’ solutions (this year or from past years)
Academic honesty (cont’d)

• We use code comparison software
  – Compares parse trees; immune to obfuscation
  – Produces color-coded all-student-pairs code comparisons

• Don’t copy code—you **will** be caught!

• Penalty for copying: automatic zero marks, referral for disciplinary action by UCL (usually leads to exclusion from all exams at UCL)
Today we’ll cover three topics

- Administrivia for 3007 Computer Systems: people, class policies

- What is “Systems?”

- Introduction to x86-64 assembly (for your first coursework, handed out today!)
What Is Systems?

• Mastery of the “whole stack” and interactions among its parts:
  – CPU hardware
  – Memory system
  – Operating system (I/O, virtual memory)
  – System software (libraries, memory allocator, compiler)
  – C program behavior

• Abstraction: limited in power
  – Ultimately, must grapple with peccadilloes of real hardware, vast platform software, programming language, and even the compiler...
Systems? (II)

• How can we achieve correctness and high performance in practice? Examples:
  – the C language rife with undefined behavior: a severe correctness (and security) threat (but a performance boon)
  – OS architecture dictates performance of application code; cannot hide completely with modularity/layering
  – even OS guarantees rely on foibles of CPU hardware...
Theoretical explanation

The Intel Optimization Reference Manual says the following regarding Sandy Bridge (and later microarchitectural revisions) in section 2.3.2.3 ("Branch Prediction"): 

Branch prediction predicts the branch target and enables the processor to begin executing instructions long before the branch true execution path is known.

In section 2.3.5.2 ("L1 DCache"): 

Loads can:

[...]

- Be carried out speculatively, before preceding branches are resolved.
- Take cache misses out of order and in an overlapped manner.


Implicit caching occurs when a memory element is made potentially cacheable, although the element may never have been accessed in the normal von Neumann sequence. Implicit caching occurs on the P6 and more recent processor families due to aggressive prefetching, branch prediction, and TLB miss handling. Implicit caching is an extension of the behavior of existing Intel386, Intel486, and Pentium processor systems, since software running on these processor families also has not been able to deterministically predict the behavior of instruction prefetch.

Consider the code sample below. If `$arr1->length` is uncached, the processor can speculatively load data from `$arr1->data[untrusted_offset_from_caller]`. This is an out-of-bounds read. That should not matter because the processor will effectively roll back the execution state when the branch has executed; none of the speculatively executed instructions will retire (e.g. cause registers etc. to be affected).
Theoretical explanation

The Intel Optimization Reference Manual says the following regarding Sandy Bridge (and later microarchitectural revisions) in section 2.3.2.3 ("Branch Prediction"):

> Branch prediction predicts the branch target and enables the processor to begin executing instructions long before the branch true execution path is known.

In section 2.3.5.2 ("L1 DCache"):

> Loads can:
> 
> 1. Be carried out speculatively, before preceding branches are resolved.
> 2. Take cache misses out of order and in an overlapped manner.


> Implicit caching occurs when a memory element is made potentially cacheable, although the element may never have been accessed in the normal von Neumann sequence. Implicit caching occurs on the P6 and more recent processor families due to aggressive prefetching, branch prediction, and TLB miss handling. Implicit caching is an extension of the behavior of existing Intel386, Intel486, and Pentium processor systems, since software running on these processor families also has not been able to deterministically predict the behavior of instruction prefetch.

• By the end of 3007, you should be prepared to comprehend Meltdown and Spectre!
C? x86-64? My Future Is In Machine Learning!

- Cutting-edge ML backed by custom TPU, unique system software and OS support...
Systems? (IV)

• Being at ease with the machine...
Handout: CW1

• Defusing a Binary Bomb
• Due start of lecture, 18th January (9 days)
• Start right away!
  – Need to attempt the CW to formulate questions to ask at office hours