CS3104 – Operating Systems

Assignment: P2 – An analysis of paging

Deadline: 23 November 2022 *Credits*: 50% of coursework mark

MMS is the definitive source for deadline and credit details

You are expected to have read and understood all the information in this specification and any accompanying documents at least a week before the deadline. You must contact the lecturer regarding any queries well in advance of the deadline.

Aim / Learning objectives

The purpose of this assignment is three-fold:

- to provide insight into a real paging architecture;
- to understand the structure of a page table in a real system; and
- to understand how virtual addresses are converted into physical addresses.

This assignment has four parts of increasing difficulty. We encourage you to tackle them in order. Your report should have four sections, one corresponding to each part.

Part 1: Page table layout

In this part, you will explore the page table for an imaginary architecture. The architecture is specified as follows:

- It is a 16-bit architecture.
- The architecture only supports single-level paging.
- It supports two protection bits: *read-only*, and *executable*.
- Page/frame size is 128 bytes.

Show (using a diagram or table in your report) the layout of the page table and show an example row. Make sure to explain how many bits are needed to store each row of the page table, and how many bytes are needed to store the entire page table. Explain the maximum size of the virtual and physical memory spaces. Explain the process of converting a virtual memory address into a physical memory address on the following three examples:

- 0xbeef
- 0xc001
- the last five numbers of your student ID, interpreted as an integer.

In your answers, explain which parts of the process are performed by the kernel and which by the hardware. State any assumptions you make.

Part 2: Research into ARM architecture

You should research and write a short essay about paging on ARM. This can be any version of the ARM architecture (e.g. Armv8-A) but you should not mix features from different versions. Please note that on the ARM architecture, page tables are commonly referred to as translation tables. Your description should not be longer than **1500 words** (advisory limit). Overlength will not be penalised, but we will reward conciseness and precision and may penalise lack of focus and lack of clarity. In your description, make sure to relate your description to concepts covered in lectures and the textbook such as (but not limited to):

- the page and frame sizes,
- the type of page table (i.e. flat, hierarchical, inverted...),

- the number of bits used to represent the frame number and offset,
- the maximum amount of physical and virtual memory,
- the amount of memory required to hold the page table,
- any protection bits stored in the page table, etc.

You can use the references provided in this spec as a starting point, but you are encouraged to research beyond this and clearly identify all the sources you used. You are encouraged to make use of diagrams as appropriate. When using diagrams, avoid screenshots of diagrams from the internet, create your own and explain them appropriately. Make sure to remember good academic practice policy when referencing your sources.

Part 3: Implementation

There is starter code on studres for simulating paging on the 16-bit architecture from Part 1. In this part, you will need to implement a simulator for the paging process. The simulator works as follows: a contiguous area of memory (store) acts as a model for physical memory. It consists of 128-byte frames. There is a page table (a large array) which defines how a virtual address is mapped to a physical frame. The implementation will comprise three main parts:

- code to allocate the memory for the page table structure (pt_init),
- code to add entries to this page table (map_page_to_frame), and
- code to simulate reading and writing to virtual addresses using provided page table and store.

Function prototypes are defined in paging.h, which should not be changed. Your implementation should go into paging.c . There is also a file called test.c which contains two very simple tests which illustrate how to use your simulator in practice. You can use this as a starting point for your own testing.

Make sure you describe your implementation in the report and justify main design decisions. There is very little actual code needed, but you will need to think carefully about how to implement the page table from Part 1, and how to use pointers appropriately to make sure your implementation is correct. You should provide evidence of testing that demonstrates that your solution works for a range of non-trivial tasks.

Part 4: Advanced topics

Only attempt this part after completing parts 1-3. In this part, you will explore an advanced paging topic and update your implementation. You will choose one of the following three topics:

- hierarchical paging, or
- inverted page table, or
- virtual memory (swapping).

You will update your implementation from Part 3 to implement your chosen extension. You should place this implementation in a separate C file with a separate build target, since it will be assessed separately from Part 3! The header file should remain the same so that you can link your tests against either version. In the report you should explain what changes were needed to your code, and explain any design decisions you had to make.

N.B. No extra credit will be given for implementing more than one of the suggested extensions or for implementing something else. Each one of these extensions is challenging enough if implemented well.

Submission

Submit a single ZIP file containing the code you developed for Parts 3-4, and a PDF document containing a report that documents your work on all four parts of the practical. The report should contain a short introduction and conclusion, and one section for each part listed above. Make sure to cite all external material and to number and caption any figures.

Resources

You may find some of these sources a good starting point, but you are encouraged to look for others:

https://developer.arm.com/documentation/102376/0100/Overview
https://link.springer.com/chapter/10.1007/978-3-319-51517-5_6
https://www.ic.unicamp.br/~celio/mc404-2013/arm-manuals/Paging%20Systems.pdf
https://wiki.osdev.org/ARM_Overview#Memory
https://www.kernel.org/doc/html/latest/arm64/memory.html

Assessment
Marking will follow the guidelines given in the school student handbook (see link in next section). Some specific descriptors for this assignment are given below:

Mark range	Descriptor
1 - 6	A submission that shows little evidence of any attempt to complete the work.
7 - 10	Successful completion of Part 1 (page table), adequately documented and explained.
11 - 13	Successful completion of Part 1 and Part 2 which shows a good level of detail and understanding, and manages to effectively relate page table layout and ARM architecture to concepts covered in lectures.
14 - 17	A submission which completes Parts 1-3 to a good standard. The page table should be correct, and the report demonstrate understanding. The description of paging on ARM should be correct, informative, and demonstrate good understanding of lecture concepts and make use of external sources. The code should be mostly correct, well-designed, and well-explained in the report.
18-20	A submission which completes Parts 1-4 to an excellent standard. It should show evidence of extensive background reading and understanding of strengths and weaknesses of paging on ARM. Part 4 should demonstrate good understanding of advanced paging concept. All work and the report must be excellent for this grade band, which includes well-tested and documented code with an insightful write-up demonstrating deep understanding of challenges and trade-offs.

Policies and Guidelines

Marking
See the standard mark descriptors in the School Student Handbook:
http://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html#Mark_Descriptors

Lateness penalty
The standard penalty for late submission applies (Scheme B: 1 mark per 8 hour period, or part thereof):
http://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html#lateness-penalties

Good academic practice
The University policy on Good Academic Practice applies:
https://info.cs.st-andrews.ac.uk/student-handbook/academic/gap.html