Institut für Informatik
Lehrstuhl für Technische Informatik

Digitale Systeme 2023

C-Programmierprojekt

1 Aufgabenstellung

Ihre Aufgabe besteht darin, einen virtuelle Speicherverwaltung in Form einer Seitenauslagerung (Paging) mit der Programmiersprache C zu entwickeln.

2 Implementierung der Seitenauslagerung

Eine virtuelle Speicherverwaltung ermöglicht es, die Adressräume eines Programms in physische Adressen zu übersetzen. Die physische Adresse kann dann für den Zugriff auf den Hauptspeicher verwendet werden. Dieser Prozess wird als Adressübersetzung bezeichnet. Heute sind die beiden Speicherhierarchieebenen, die über virtuellen Speicher gesteuert werden, mit DRAM und Festplatten oder Flash Speicher realisiert. Mittels einer Seitentabelle (Page Table) kann für jede virtuelle Adresse die korrespondierende physische Adresse erstellt werden.
Der virtuelle Adressraum wird in eine Reihe von Seiten gleicher Größe aufgeteilt. Die Seitengröße ist immer eine Potenz von 2. Der physische Adressraum wird ebenso aufgeteilt, wobei die Größe der einzelnen Teile die gleiche ist wie bei den Seiten, sodass jeder Teil des Hauptspeichers genau eine Seite aufnehmen kann. Diese Teile des Hauptspeichers, die Seiten aufnehmen, heißen Seitenrahmen (Page Frames). Die Abbildung von einer virtuellen Adresse auf eine physische Hauptspeicheradresse übernimmt die Speicherverwaltungseinheit (MMU für Memory Management Unit). Für dieses C-Programmierprojekt soll die MMU eine virtuelle 32 Bit Adresse auf eine 16 Bit Adresse abbilden. Die Seiten sollen immer eine Größe von 4 KB haben. Wenn die MMU eine virtuelle 32-Bit-Adresse erhält, trennt sie die Adresse in eine 20 Bit große virtuelle Seitennummer und in einen 12-Bit großen Offset (der 12-Bit Offset ergibt sich aus $2^{12} = 4$ K). Die Seitennummer ist der Index für die Seitentabelle, die somit 2^{20} Zeilen hat. Die Abbildung 1 zeigt das beschriebene Vorgehen. Abweichend von der Abbildung 1 hat die Seitentabelle folgende Spalten:

1. Der Seitenrahmen, der die hochwertigen Bits (MSBs) der pysikalischen Adresse beinhaltet
2. Ein Bit, das anzeigt, ob sich die Seite im Arbeitsspeicher befindet

3. Ein Bit, das anzeigt, ob etwas innerhalb der Seite verändert wurde

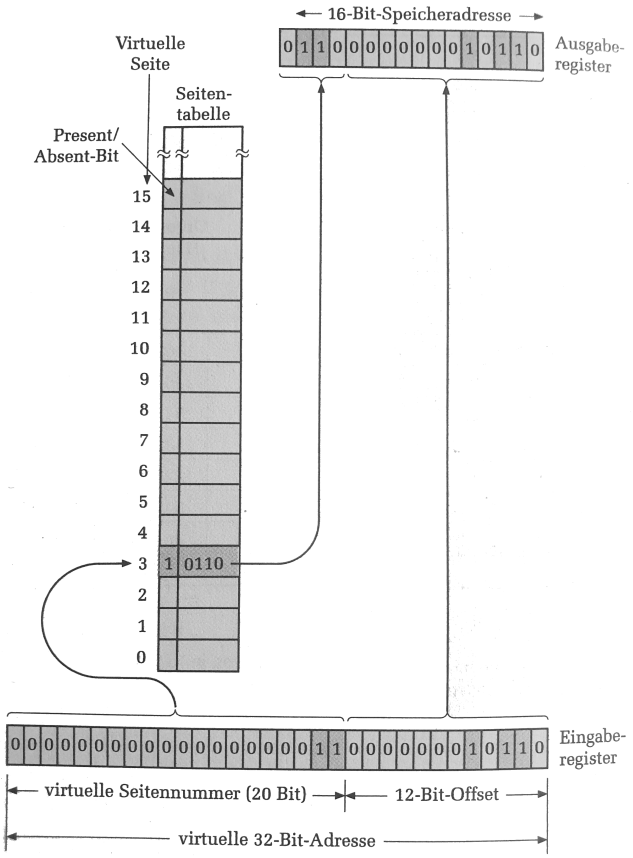

Abbildung 1: Quelle: in Anlehnung an Computerarchitektur, Tanenbaum, 2006, Seite 464
Bildung einer Hauptspeicheradresse aus einer virtuellen Adresse

Wenn der Arbeitsspeicher voll ist, aber eine weitere Seite angefragt wird, muss eine andere Seite ausgelagert werden. Wenn etwas in dem Adressbereich der auszulagernden Seite verändert wurde, also das „dirty bit“ gesetzt ist, muss diese wieder zurück auf das langsamere Medium geschrieben werden. Die Strategie zur Auswahl der auszulagernden Seite ist Ihnen überlassen. Eine einfache Strategie wäre der LRU (Least Recently Used) Algorithmus. Eine deutlich umfangreichere Erklärung zu virtuellen Speicher finden Sie in:

- Computer Organization and Design - The Hardware Software Interface, D. A. Patterson, J. L. Hennessy, 2021, 6 Aufl. Seite 440 bis 463
- Computerarchitektur, A. S. Tanenbaum, 2006, 5. Aufl. Seite 459 bis 469
- Folien zur Vorlesung Digitale Systeme, Dr. Sommer, 2023, DS_09_speicherverwaltung.pdf ab Seite 43

3 Vorgehen

In der Datei `main.c` gibt es bereits einige Hilfsfunktionen, die Sie zur Lösung Ihres virtuellen Speichers verwenden können. Es ist Ihnen überlassen, ob Sie die Funktionen in der Form beibehalten oder eine eigenständige Struktur erstellen (außer die Funktion get_data). Um das Programmierprojekt erfolgreich abzuschließen, müssen Sie folgende Funktionalität implementieren:

- Übersetzung einer virtuellen in eine physikalische Adresse
- Auslagern einer Seite
- Schreiben einer Seite vom Hauptspeicher (ra_mem) in den langsameren Speicher (hd_mem)
- Eine Seite vom langsameren Speicher in den Arbeitsspeicher schreiben
- Eine Funktion, die eine virtuelle Adresse als Übergabeparameter erhält und den Wert, der an dieser Adresse gespeichert ist, zurückgibt. **Es ist nicht erlaubt, direkt hd_mem zu adressieren!** Diese Funktion muss mit

  ```
  uint8_t get_data(uint32_t virt_address){...}
  ```

  definiert werden. Halten Sie sich an Namen, Parameter und Rückgabewert.
- Eine Funktion zum Setzen eines Wertes an einer virtuellen Adresse. **Auch hier ist es nicht gestattet, direkt hd_mem zu adressieren!**

Die Programmiersprache C besitzt keine direkte Möglichkeit einzelne Bits zu verändern, es gibt aber einen Trick, die eine Manipulation ermöglicht. Damit die relevanten Bits richtig interpretiert werden, müssen diese an die richtige Stelle verschoben werden. Dafür ist in der Programmiersprache C der Schiebeoperator „>>“ vorgesehen. Um unrelevante Bits einer Variablen zu entfernen, kann die Variable mit einem Bitmuster „&“ verknüpft werden.

4 Abgabe

main.c

Ihre komplette Lösung soll in der main.c Datei von Ihnen programmiert werden. Die Verwendung weiterer Quelldateien in Abgaben ist nicht zulässig und steht nicht zur Diskussion. Weiterhin dürfen ausschließlich die in der Aufgabenstellung bereits inkludierten Header verwendet werden, die Verwendung weiterer Header ist nicht zulässig. Das Programm soll mit dem Befehl

```
gcc main.c -o hu_soft_mmu -std=c11
```

übersetzt werden können. Als Referenzsystem dient `gruenau4.informatik.hu-berlin.de`. Auf diesem Rechner muss Ihr Code funktionieren. Abgaben, die dort nicht kompilieren oder dort nicht korrekt ausgeführt werden können, werden als nicht bestanden gewertet.

Die Aufgabe gilt als bestanden, wenn Ihr Programm zum einen sämtliche Testfälle in der main() zufriedenstellend ausführt und zum anderen die Funktion des virtuellen Speichers in der Form einer Seitenauslagerung erfüllt.

5 Zeitplan

Sie können sofort mit der Bearbeitung der Aufgabe beginnen. Bis zur finalen Abgabefrist haben Sie beliebig viele Abgabeversuche, eine erste Abgabe ist ab sofort möglich. Wir bemühen uns, Ihnen nach jedem Abgabeversuch kurzfristig Rückmeldung zu geben, ob Ihre Lösung korrekt ist oder falls nicht, inwiefern sich Ihr Programm falsch verhält. **Nach der finalen Abgabefrist werden keinerlei weitere Abgaben mehr berücksichtigt.** Wir ermutigen Sie ausdrücklich, so früh wie möglich mit der Bearbeitung der Aufgabe zu beginnen und einen ersten sinnvollen Lösungsversuch bis Anfang August abzugeben. Je nach Aufkommen von Abgaben können wir wenige Wochen vor der finalen Deadline nicht mehr garantieren, kurzfristig individuelles Feedback zu geben, das Ihnen auch wirklich weiter hilft. Da eine vollständig korrekte Lösung für das Bestehen erforderlich ist, sind erfahrungsgemäß fast immer mehrere Abgaben notwendig[1].

Wenn Sie eine korrekte Lösung eingereicht haben, müssen Sie uns für die Erbringung der Modulteilleistung zusätzlich noch im Rahmen eines kurzen persönlichen Abtestats Ihre Lösung erläutern und Fragen dazu beantworten. Die Abtestate werden online über Zoom stattfinden.

Datum	Beschreibung
07.08.2023	Empfohlene Deadline für eine erste Abgabe; bei Abgabe bis zu diesem Termin können wir im Falle von Problemen mit der Abgabe gewährleisten, dass nach einer ersten Rückmeldung noch Zeit zur Nachbesserung bis zur endgültigen Deadline bleibt.
01.09.2023 (23:59 Uhr MESZ)	Spätestmögliche Deadline für die Abgabe der Endversion, die alle Kriterien vollständig erfüllen muss.
01.08.2023 – 15.09.2023	In diesem Zeitraum finden die mündlichen Testate in Einzelsitzungen statt. **Eine frühere Abnahme ist nach individueller Absprache möglich und sehr willkommen!**

Viel Erfolg!

[1] Die Deadlines sind ernst gemeint. Von einer Abgabe ein paar Stunden, Minuten oder Sekunden vor der finalen Abgabefrist raten wir dringend ab. Technische Probleme bei der Abgabe sind kein Grund für eine Ausnahme von der Frist.