HOCHSCHULE RUHR WEST
UNIVERSITY OF APPLIED SCIENCES
INSTITUT INFORMATIK

Betriebssysteme

SS 2022

Dr.-Ing. Ahmad Rabie
ahmad.rabie@hs-ruhrwest.de

Aufgabe P3: Speicherverwaltung

Gegeben sind die folgenden Strukturen:

```
struct PageTableEntry
{
    unsigned int page_frame_index;
    unsigned char frame_attributes;
};

struct PageTable
{
    struct PageTableEntry** entries;
    unsigned int size;
};

struct TLBEntry
{
    unsigned int page_index;
    unsigned int page_frame_index;
    unsigned char frame_attributes;
};

struct TLB
{
    struct TLBEntry** entries;
    unsigned int size;
};

struct Process
{
    unsigned char process_id;
    struct PageTable* page_table;
};
```

```
struct MMU
{
    struct TLB* tlb;
    struct Process* process;
};


struct ED_209
{
    struct MMU* mmu;
};
```

In diesem Versuch geht es um einen Roboter des Typs ED-209. Bei der ersten Vorführung dieser Maschine sind leider einige Dinge schief gelaufen. Jedoch haben die Ingenieure die Fehlerquelle sehr schnell ausfindig gemacht: Es lag an einer fehlerhaften MMU in der 24-Bit-CPU des Roboters. Aus Zeitmangel konnten die Ingenieure den Fehler nicht beheben. Stattdessen schrieben sie einen Zusatzabschnitt in das Benutzerhandbuch, da der Fehler leicht von Benutzern mit elementaren C- und Betriebssystem-Kenntnissen behoben werden kann.

"ED-209" arbeitet mit 24-Bit Adressen. Bevor ein Programm ausgeführt wird, werden alle Instruktionen ausgelesen und in den TLB der CPU kopiert. Da die Programme sehr kurz sind können sie in der Regel vollständig im Speicher behalten werden. Dennoch ist der virtuelle Adressraum von ED-209 24 Bit groß, das System selbst verfügt aber nur über 1 MegiByte an RAM, eine Page ist 1 KibiByte groß. Der Kontrollverlust über ED-209 während der ersten (und letzten) Vorführung, lag an einem fehlerhaften Eintrag im TLB. Das Fehlverhalten kann durch die folgenden Schritte leicht korrigiert werden.

1. Bestimmen Sie die folgenden Größen und legen Sie diese als Konstanten (z. B. als `defines`) im Programm fest: Größe einer Page, Größe des Systemspeichers, Größe des virtuellen Speichers für einen Prozess, Anzahl der Pages, Anzahl der Page Frames sowie Größe einer Page Frame. Schreiben Sie anschliessend eine Funktion `void printSystemInfo()` welche diese Größen ausgibt.

2. Schreiben Sie jeweils Funktionen `struct PageTable* createPageTable()`, `struct TLB* createTLB(unsigned int size)` und `struct Process* createProcess(unsigned char id)` welche eine `PageTable` bzw. einen `TLB` bzw. einen Prozess erstellen.

3. Schreiben Sie eine Funktion `unsigned int countTLBEntries(struct Process* p)` welche die Einträge in der `PageTable` von Prozess `p` zählt, welche das Flag `FRAME_TLB` besitzen.

4. Schreiben Sie eine Funktion `void copyTLBEntries(struct Process* p, struct TLB* tlb)` welche alle Einträge der `PageTable` von `p` mit `FRAME_TLB` Flag in den TLB `tlb` kopiert.

5. Schreiben Sie eine Funktion `unsigned int translate(unsigned int virtual_address, struct PageTable* pt)` welche eine virtuelle Adresse mit Hilfe der `PageTable` in eine physikalische Adresse übersetzt. Hierbei dürfen Sie davon ausgehen, dass die

entsprechende Page bereits im RAM liegt, d. h. Sie müssen kein *swapping* nachbilden. Jedoch soll für jede angefragte Page der entsprechende Eintrag in der `PageTable` mit den Flags `FRAME_TLB` sowie `FRAME_REFERENCED` versehen werden. Verwenden Sie ggf. globale Variablen.

6. Schreiben Sie eine Funktion `int preprocessInstructions(struct MMU* mmu, unsigned int* instructions, unsigned int instruction_count)` welche die Instruktionen aus `instructions` liest und in physikalische Adressen umrechnet. Die Funktion soll außerdem alle referenzierten `PageTable` Einträge in den `TLB` der `MMU` kopieren. Falls keine Einträge referenziert wurden, so soll diese Funktion den Wert `-1` zurückgeben, andernfalls den Wert `0`.

7. Nutzen Sie die vorhandene Funktion `void startED209(struct ED_209* ed)` um einen virtuellen `ED_209` mit dem Programm

   ```
   unsigned int edSimulation[5] = { 0xBDDE10, 0x5FFFE, 0x54AC01, 0x540C,
      0xFFFFE };
   ```

   zu testen. Hierbei müssen Sie zuvor natürlich einen `ED_209` erstellen und diesen mit einer `MMU` versorgen, welche einen korrekt gefüllten `TLB` besitzt. Sollte Ihre Implementierung korrekt sein, so erhalten Sie die Ausgabe:
   `"ED-209 is working within normal parameters -> SIMULATION SUCCEEDED"`.

Mehr Details entnehmen Sie der Abbildung auf der letzten Seite der Datei.

Abgabe muss bis spätestens 18.05.2022 erfolgen.