RTX Software Design Report

Clement Hoang

20531116
c8hoang@uwaterloo.ca

David Su

20516776
dysu@uwaterloo.ca

Cole Vander Veen

20503626
cgvander@waterloo.ca

Peter Li

XXXXXXXXX
y648li@uwaterloo.ca

Winter 2016

Contents

List of Algorithms

List of Figures

Chapter 1

Introduction

The purpose of this report is to outline the design of the RTX written by the group members, Clement Hoang, David Su, Peter Li, and Cole Vander Veen, as part of the SE350 course at the University of Waterloo. The OS is designed for a Keil MCB1700 Cortex-M3 board, with a LPC1768 microcontroller.
It is aimed to provide documentation for the operating system, in order to facilitate the use and understanding for anyone interested in programming for the OS. As such, this report outlines the global variables used in the OS, and then moves on to describing the kernel API in a modular and chronological way, from when we implemented it. Finally, the report closes with some analysis on the OS, and challenges that the group faced for the duration of the lab.

Chapter 2

Design Description

2.1 Global Variables and Data Structures

- `memQueue`: A data structure that models the free physical memory in the OS, by splitting the heap into blocks of equal size. It is represented by a `MemQueue` data structure, which is a linked list of `MemBlock` nodes of size `BLOCK_SIZE`. It is used by the kernel API when releasing and requesting memory, by popping a block when it is used by a process, and pushing it back in when it is released.
  - `MemBlock`: To expand, the `MemBlock` is a C-struct that holds a pointer to the next `MemBlock` in the queue. It also has reserved space in the front in case the block needs to hold an envelope
- `gp_pcbs`: A pointer to an array of `PCB` structs. It holds the state of all the process control blocks that are in the OS, and is interacted with by functions that change and read PCB states. For example, setting the process priority or getting the process priority uses `gp_pcbs` to access the priority of a specific PCB.
  - `PCB`: a model of a process and its state. The `PCB` contains the following fields:
    - `mp_sp`: stack pointer of the process
    - `m_pid`: ID of the process
    - `m_priority`: priority of the process
    - `m_state`: state of the process
    - `nextPCB`: pointer to the next `PCB`, if it is in a queue
    - `msgHead`: beginning of the message queue
    - `msgTail`: end of the message queue
- `gp_stack`
- `p_end`
- `numOfBlocks`

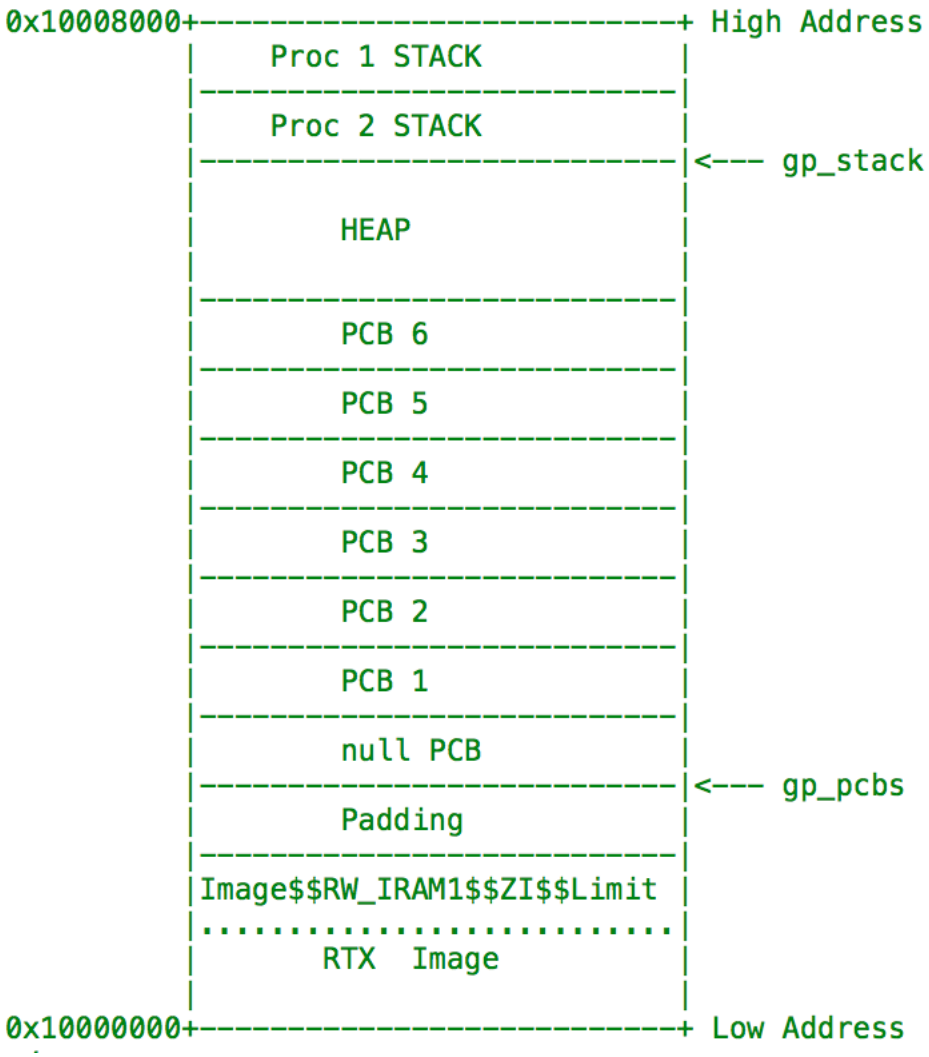

Figure 2.1: Memory Layout

2.2 Memory Management

2.2.1 Memory Structure

dsfdasfdsafdsafdsafsadfdsaf

2.2.2 Requesting Memory Blocks

```
int k_request_memory_block(void);
```

describe input, output, effects

2.2.3 Releasing Memory Blocks

```
int k_release_memory_block(void* memory_block);
```

describe input, output, effects

Algorithm 1 k_request_memory_block

```
1: procedure REQUEST_MEMORY_BLOCK
2:     while heap is full do
3:         block the current process
4:     end while
5:     update the free space list
6:     return the address of the top of the block
7: end procedure
```

Algorithm 2 The memory release function

```
1: procedure RELEASE_MEMORY_BLOCK(*memory_block)
2:     if this block is the top block of the heap then
3:         modify heap header node (never gets overwritten)
4:     end if
5:     if there is free space immediately beneath this block then
6:         combine them by increasing this block's length
7:     else this block becomes a new block node, is added to the list
8:     end if
9:     if there is free space immediately beneath this block then
10:        combine them by increasing this block's length
11:    end if
12:    if a process is blocked on memory then
13:        unblock that process, release the processor
14:    end if
15: end procedure
```

2.3 Processor Management

2.3.1 Process Control Structures

DFASFAFD

2.3.2 Process Queues

fsadfasdfadsf

2.3.3 Process Scheduling

sdfasdfasdfdasf

2.4 Process Priority Management

2.4.1 Get Process Priority

asdfadsfasf

2.4.2 Set Process Priority

dsfasdfasfsfdf

2.5 Interprocess Communication

2.5.1 Message Structure

dsfadsfadsfdasfdafs

2.5.2 Sending Messages

adsfdsafasdfasf

2.5.3 Receiving Messages

dsfafasfdasf

2.5.4 Delayed Send

sdfasfasfd

2.6 Interrupts and I-Processes

2.6.1 UART I-Process

dsfadsfadsfadsf

2.6.2 Timer I-Process

sdfasfdafd

2.7 System Processes

2.7.1 Null Process

sdfdasfafadsf

2.7.2 CRT Process

sdfdsfafaf

2.8 User Processes

2.8.1 Wall Clock Process

sdfasdfafadf

2.8.2 Set Priority Process

dsfasdfasdfadsf

2.8.3 Stress Test Processes

dfdasfasdfads

2.9 Initialization

dasfasfasfd

2.10 Testing

dfadsfasdf

2.11 Major Design Changes

dsfdafadsf

Chapter 3

Lessons Learned

3.1 Source Control and Code Management

sdfdsafsadf

Chapter 4

Team Dynamics and Individual Responsibilities

4.1 adsfadsf

dfasfasdf

Chapter 5

Timing Analysis