SE350 RTX LAB 1 – 1

P1A Fixed-Size Memory Pool Management

Irene Yiqing Huang
Electrical and Computer Engineering Department

Copyright by Yiqing Irene Huang 2022. All rights reserved

Reading List

- Lab Manual

Section	Topics	
2.1	The API	Study
2.2	Memory Management	Study
3.1.1	The Null Process	Study
3.3.1	User-Level Test Processes	Study
4	RTX Initialization	Study
6.2	Memory Management FAQ	Skim
6.5	Interrupts FAQ	Skim

- OS: Three Easy Pieces Chapter 17
  - A good reference. Section “Embedding A Free List” is most relevant.
  - Not a required reading

Copyright by Yiqing Irene Huang 2022. All rights reserved

P1 Basic Requirements & Assumptions

Basic Requirements

- Memory management (i.e., request/release memory blocks)
- Multiprogramming (processes)

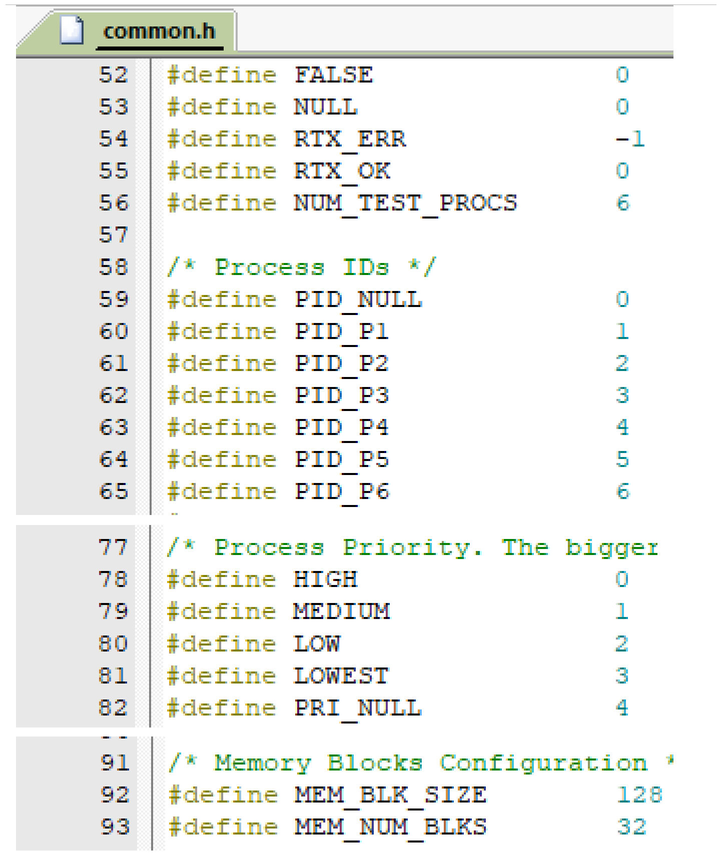
P1-A

- Pre-emptive and fixed priority scheduling

P1-B

Assumptions

- All processes are known
  - created at start-up; know each other
- Processes are non-malicious



```
#define FALSE              0
#define NULL               0
#define RTX_ERR           -1
#define RTX_OK             0
#define NUM_TEST_PROCS     6

/* Process IDs */
#define PID_NULL           0
#define PID_P1             1
#define PID_P2             2
#define PID_P3             3
#define PID_P4             4
#define PID_P5             5
#define PID_P6             6

/* Process Priority. The bigger
#define HIGH               0
#define MEDIUM             1
#define LOW                2
#define LOWEST             3
#define PRI_NULL           4

/* Memory Blocks Configuration *
#define MEM_BLK_SIZE       128
#define MEM_NUM_BLKS       32
```

Copyright by Yiqing Irene Huang 2022. All rights reserved

P1 Requirements : User API

- Memory Management: a memory pool which has fixed size of memory block and fixed number of memory blocks.

```
void *request_memory_block()
int  release_memory_block(void *memory_block)
```

P1-A

- Processor Management

```
int release_processor()
```

- Process Priority Management

```
int set_process_priority(int process_id, int priority)
int get_process_priority(int process_id )
```

P1-B

Copyright by Yiqing Irene Huang 2022. All rights reserved

Atomicity Concepts

- RTOS primitives must execute *indivisibly*
- define private kernel function

```
atomic( on / off )
```

Conceptual Only

  - atomic(on) : enables atomicity
    first executable statement in each primitive
  - atomic(off): disables atomicity
    last executable statement in each primitive before 'RET'
  - Often code in assembly

Copyright by Yiqing Irene Huang 2022. All rights reserved

P1 Requirements: Processes

- Null Process
  - A system process which does nothing in an infinite loop. PID=0.
- Test Processes
  - Up to six test processes with PIDs = 1,2, …, 6
  - User level processes, only calls the user APIs
- Initialization
  - Memory, system processes and user processes

All processes never terminate!
No new process created on the fly.

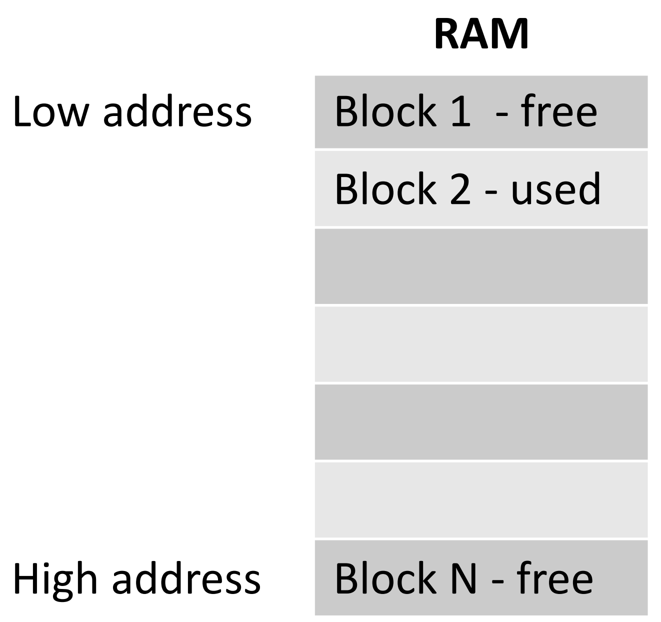
Copyright by Yiqing Irene Huang 2022. All rights reserved

Memory Pool – P1A

Copyright by Yiqing Irene Huang 2022. All rights reserved

Simple Memory Management

- A logical division of the main memory
- Main activity
  - Keep track of which blocks are free
- Design Constraints
  - Using the memory it manages for the meta-data.
  - Handling arbitrary request sequences
  - Making immediate responses to requests
  - Not modifying allocated blocks



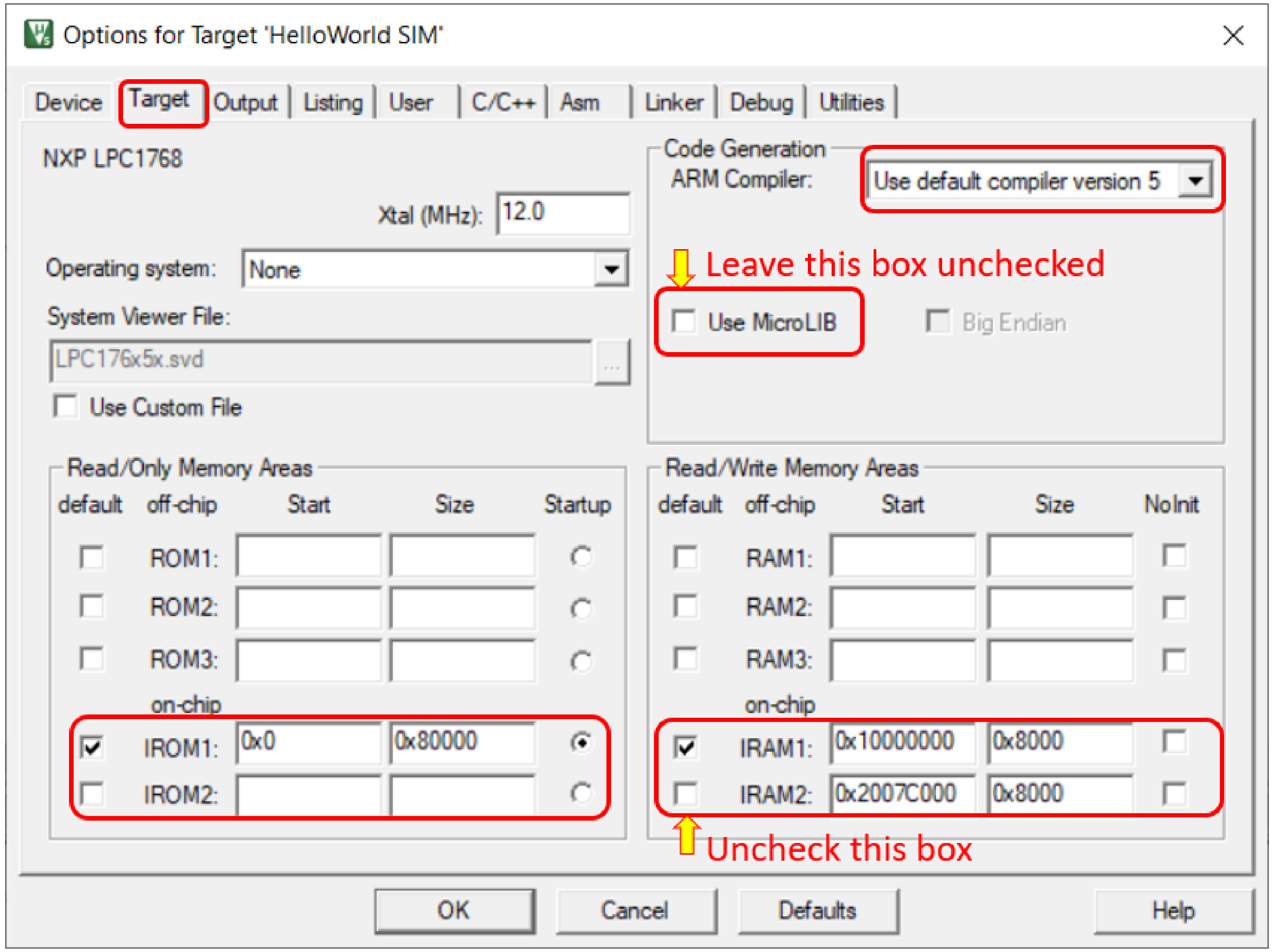
Copyright by Yiqing Irene Huang 2022. All rights reserved

LPC1768 Memory Map

0x2008 4000		
0x2007 C000	32 KiB	AHB SRAM (2 blocks of 16 KiB) (IRAM2)
0x1FFF 2000		Reserved
0x1FFF 0000	8 KiB	Boot ROM
0x1000 8000		Reserved
0x1000 0000	32 KiB	Local SRAM (IRAM1)
0x0008 0000		Reserved
0x0000 0000	512 KiB	On-chip flash

Copyright by Yiqing Irene Huang 2022. All rights reserved

SIM Target Memory Area Configuration



Options for Target 'HelloWorld SIM'

NXP LPC1768

Xtal (MHz): 12.0

Operating system: None

System Viewer File: LPC176x5x.svd

☐ Use Custom File

Code Generation

ARM Compiler: Use default compiler version 5

Leave this box unchecked

☐ Use MicroLIB ☐ Big Endian

Read/Only Memory Areas

default	off-chip	Start	Size	Startup
☐	ROM1:			○
☐	ROM2:			○
☐	ROM3:			○
	on-chip			
☑	IROM1:	0x0	0x80000	●
☐	IROM2:			○

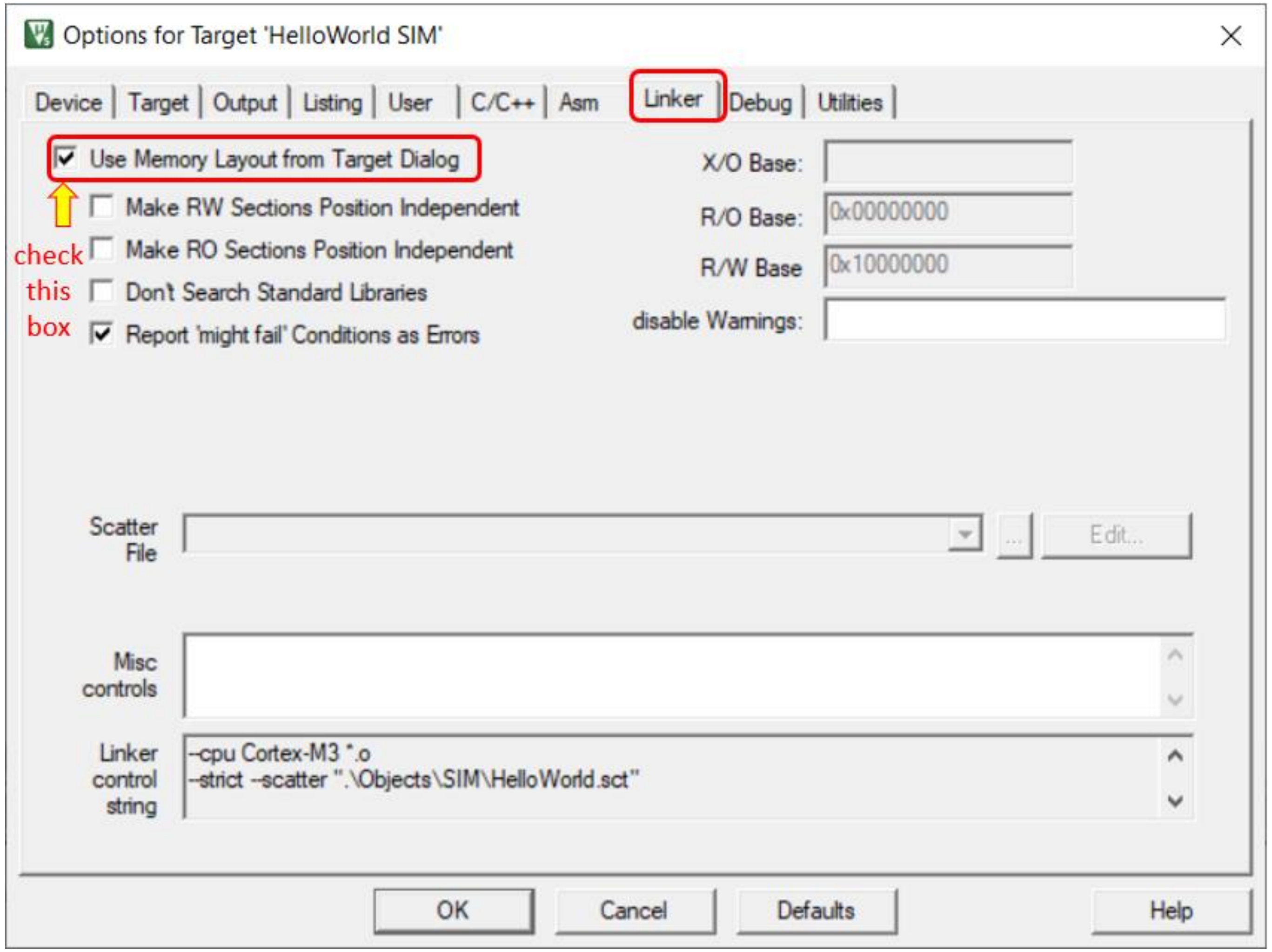
Read/Write Memory Areas

default	off-chip	Start	Size	NoInit
☐	RAM1:			☐
☐	RAM2:			☐
☐	RAM3:			☐
	on-chip			
☑	IRAM1:	0x10000000	0x8000	☐
☐	IRAM2:	0x2007C000	0x8000	☐

Uncheck this box

Copyright by Yiqing Irene Huang 2022. All rights reserved

Linker Configuration



- [✓] Use Memory Layout from Target Dialog
- [ ] Make RW Sections Position Independent
- [ ] Make RO Sections Position Independent
- [ ] Don't Search Standard Libraries
- [✓] Report 'might fail' Conditions as Errors

check this box

X/O Base:	
R/O Base:	0x00000000
R/W Base	0x10000000
disable Warnings:	

Scatter File

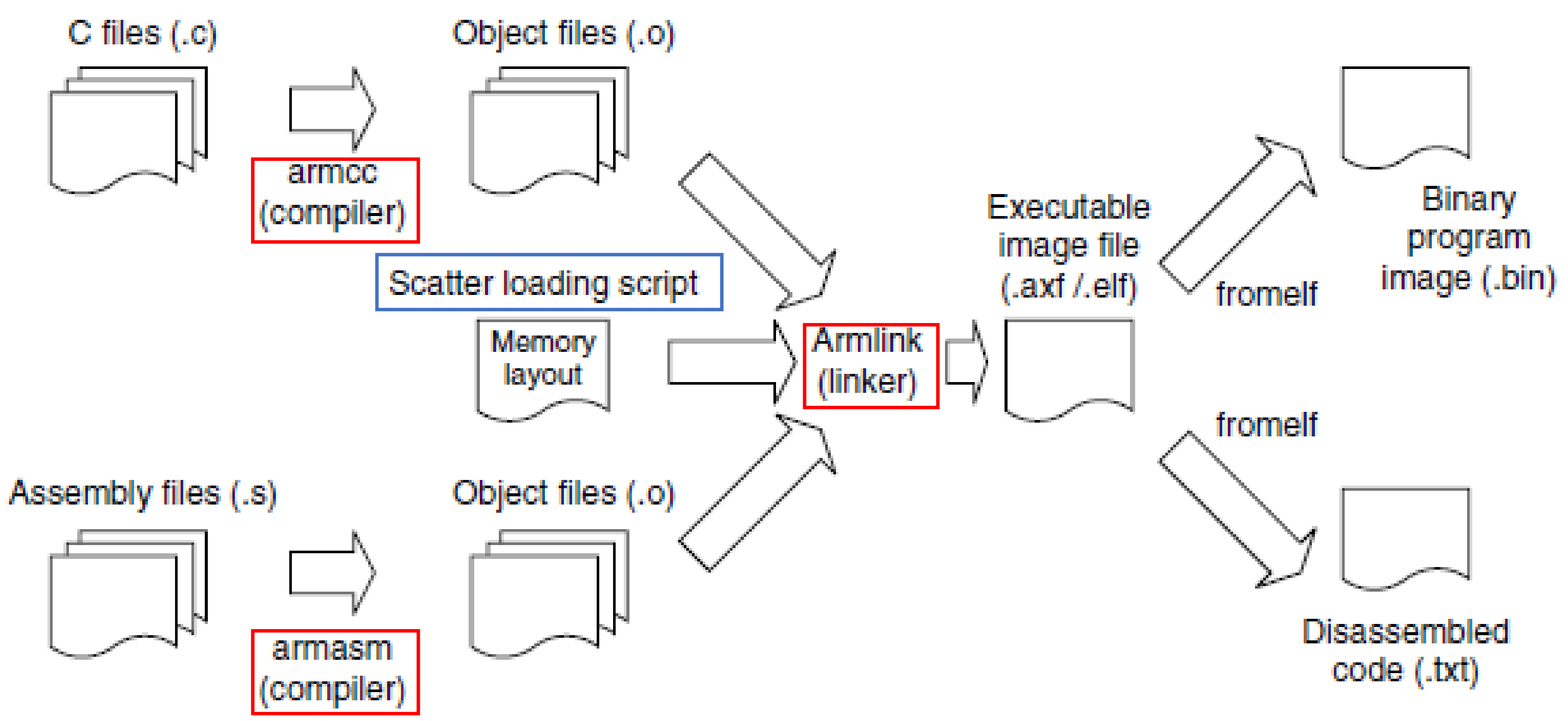
Misc controls

Linker control string

```
--cpu Cortex-M3 *.o
--strict --scatter ".\Objects\SIM\HelloWorld.sct"
```

Copyright by Yiqing Irene Huang 2022. All rights reserved

Example Flow Using ARM Development Tools

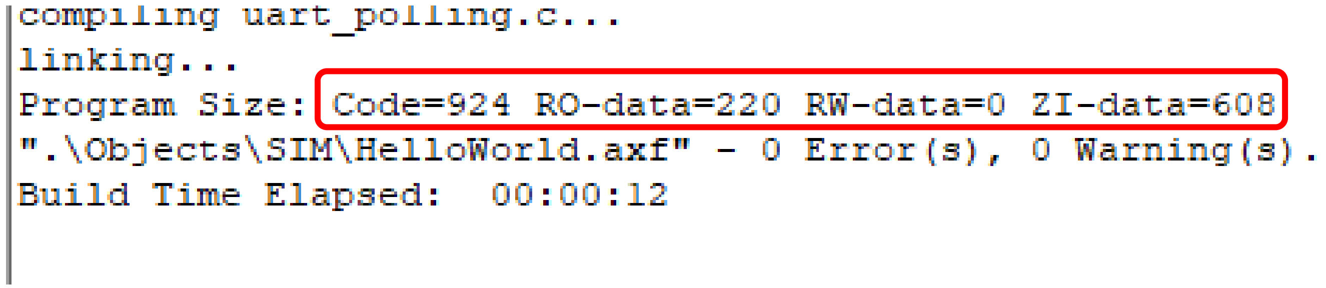


(Image Courtesy of [1])

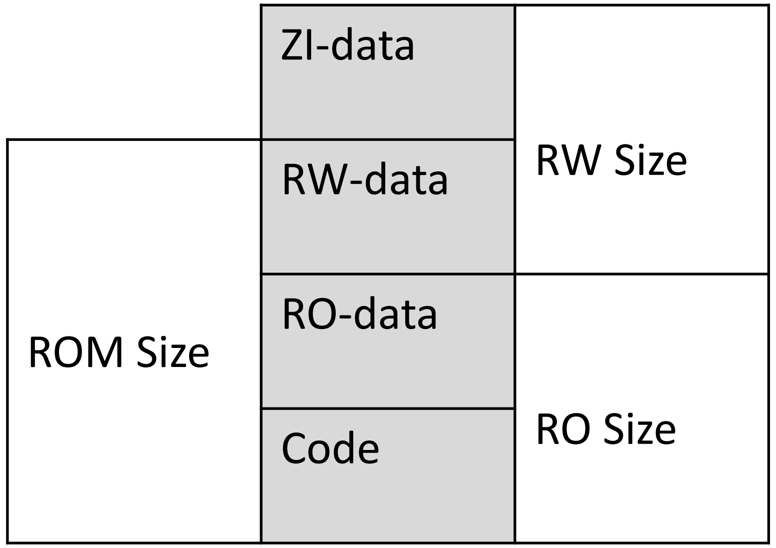
Copyright by Yiqing Irene Huang 2022. All rights reserved

Image Memory Layout

- A simple image consists of:
  - read-only (RO) section
    - Code
    - RO-data
  - Read-write (RW) section
    - a read-write data section (RW-data)
    - a zero-initialized data section (ZI-data)



```
compiling uart_polling.c...
linking...
Program Size: Code=924 RO-data=220 RW-data=0 ZI-data=608
".\Objects\SIM\HelloWorld.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:12
```

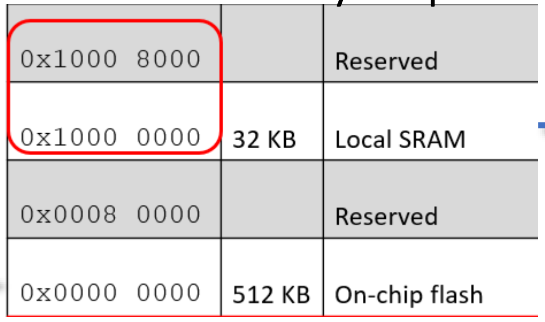


Copyright by Yiqing Irene Huang 2022. All rights reserved

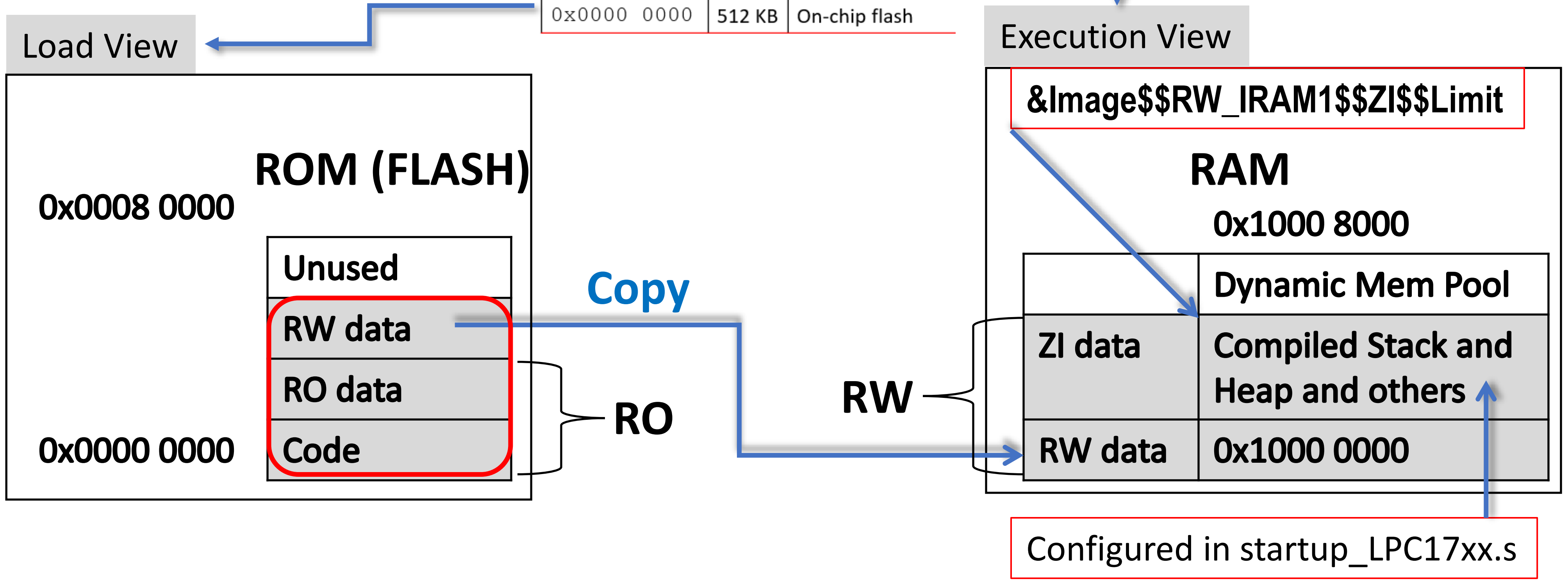
Image Execution View

- The SIM Target

LPC1768 Memory Map



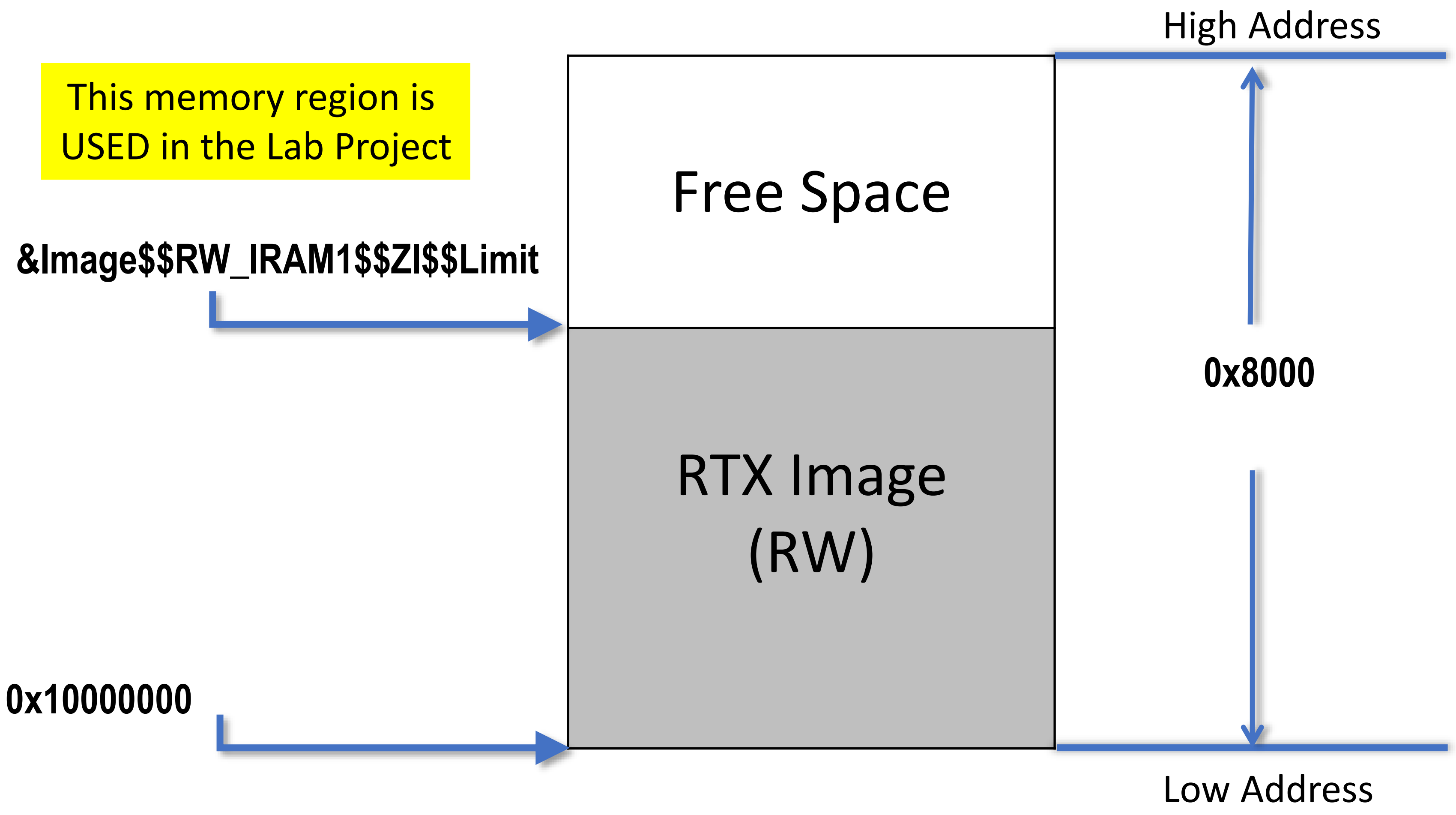
0x1000 8000		Reserved
0x1000 0000	32 KB	Local SRAM
0x0008 0000		Reserved
0x0000 0000	512 KB	On-chip flash



Copyright by Yiqing Irene Huang 2022. All rights reserved

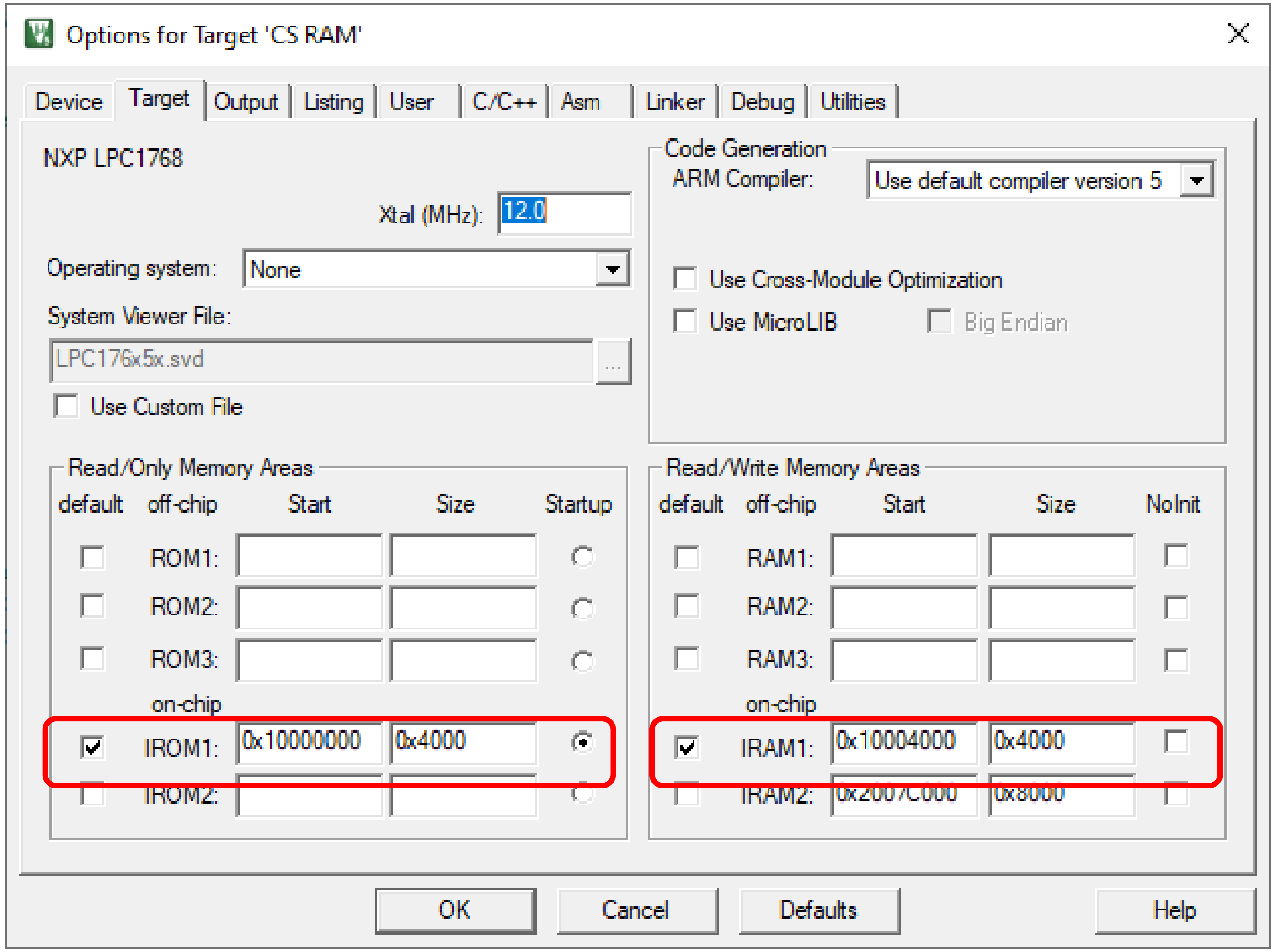
IRAM1 Memory Execution View – SIM Target

This memory region is USED in the Lab Project



Copyright by Yiqing Irene Huang 2022. All rights reserved

RAM Target Memory Area Configuration



Options for Target 'CS RAM'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities

NXP LPC1768

Xtal (MHz): 12.0

Operating system: None

System Viewer File: LPC176x5x.svd

☐ Use Custom File

Code Generation

ARM Compiler: Use default compiler version 5

☐ Use Cross-Module Optimization

☐ Use MicroLIB ☐ Big Endian

Read/Only Memory Areas

default	off-chip	Start	Size	Startup
☐	ROM1:			○
☐	ROM2:			○
☐	ROM3:			○
	on-chip			
✓	IROM1:	0x10000000	0x4000	●
☐	IROM2:			○

Read/Write Memory Areas

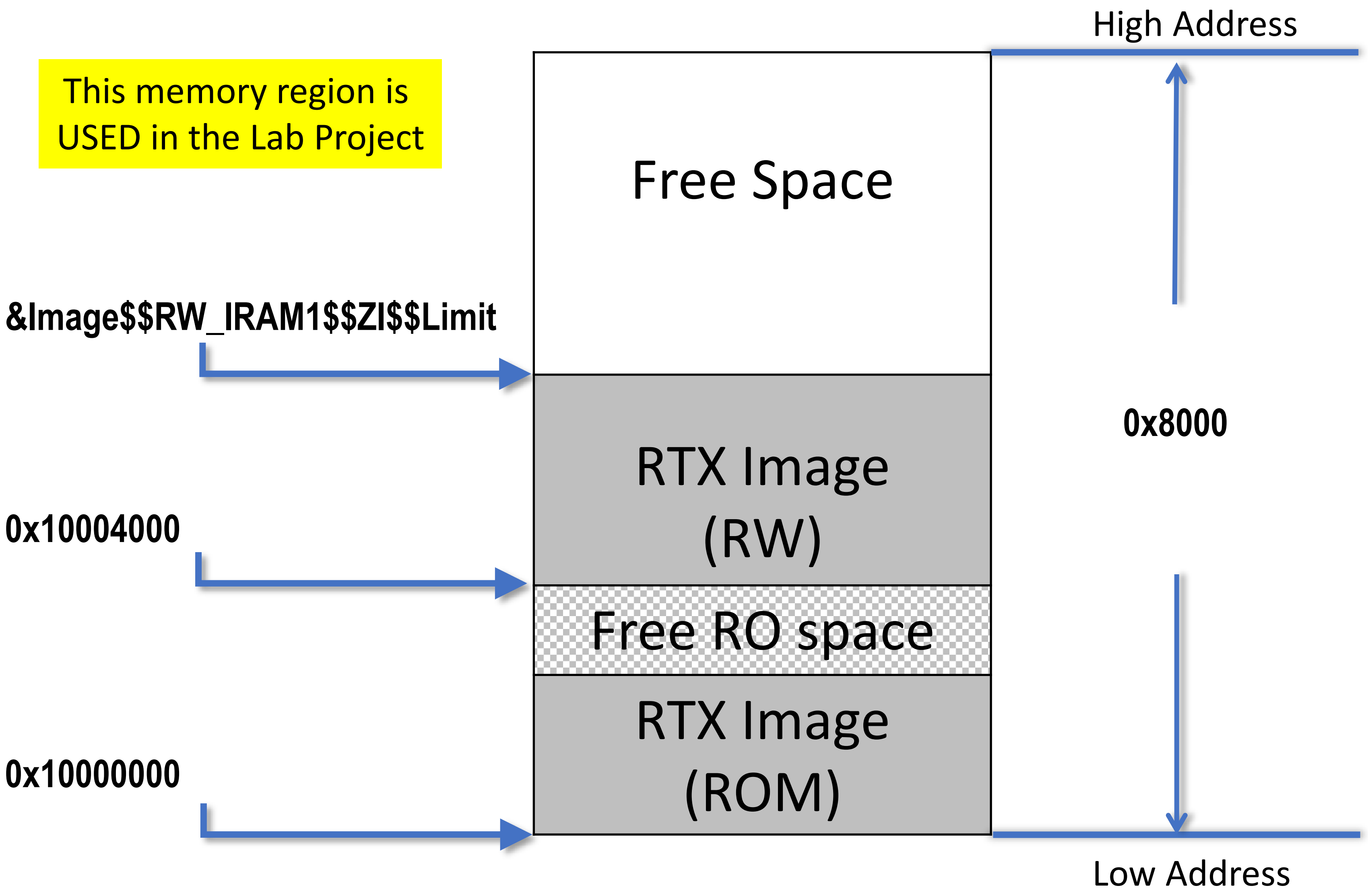
default	off-chip	Start	Size	NoInit
☐	RAM1:			☐
☐	RAM2:			☐
☐	RAM3:			☐
	on-chip			
✓	IRAM1:	0x10004000	0x4000	☐
☐	IRAM2:	0x2007C000	0x8000	☐

OK | Cancel | Defaults | Help

Copyright by Yiqing Irene Huang 2022. All rights reserved

IRAM1 Memory Execution View – RAM Target

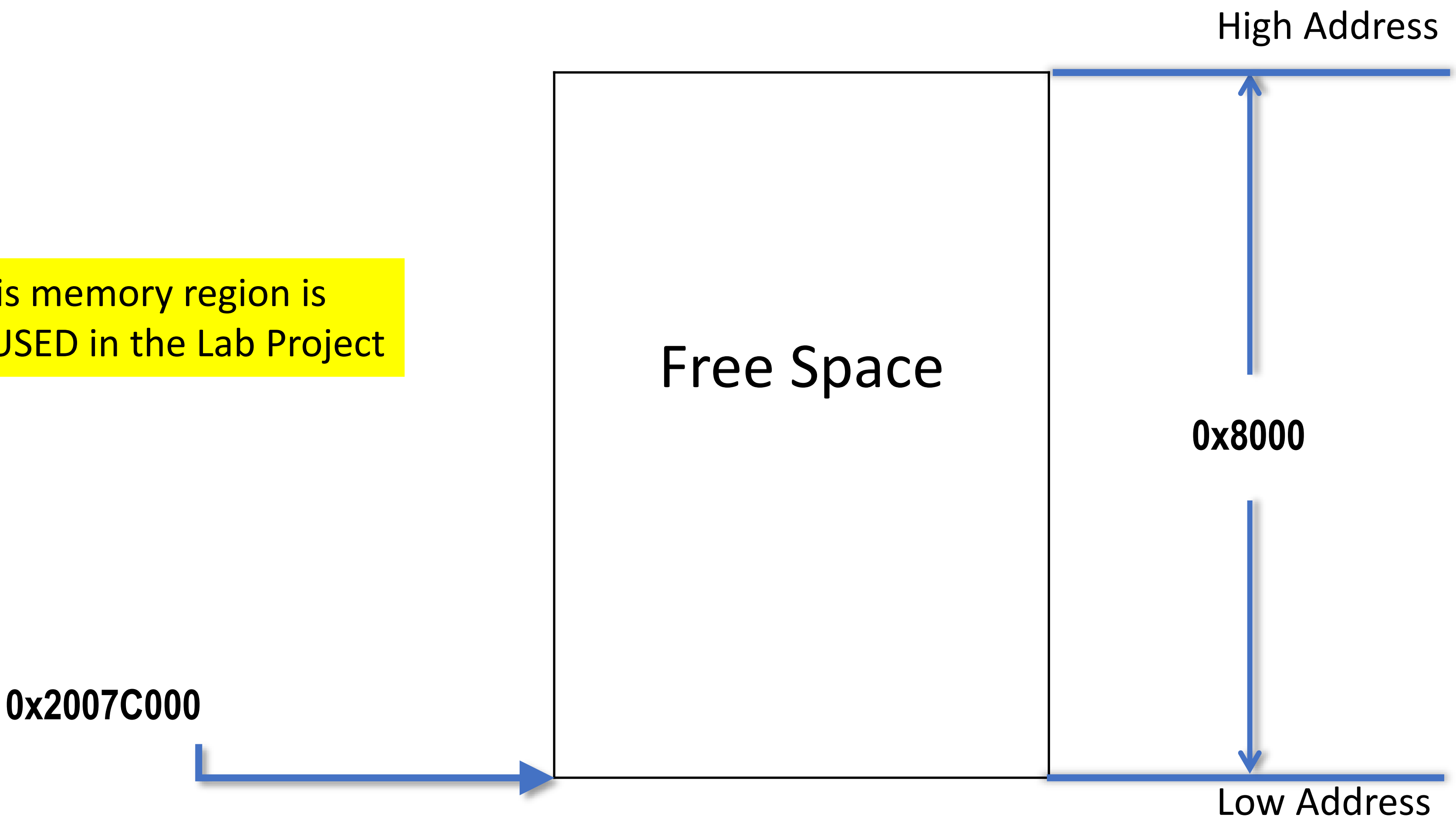
This memory region is USED in the Lab Project



Copyright by Yiqing Irene Huang 2022. All rights reserved

IRAM2 Memory Execution View

This memory region is NOT USED in the Lab Project



Copyright by Yiqing Irene Huang 2022. All rights reserved

End Address of the Image

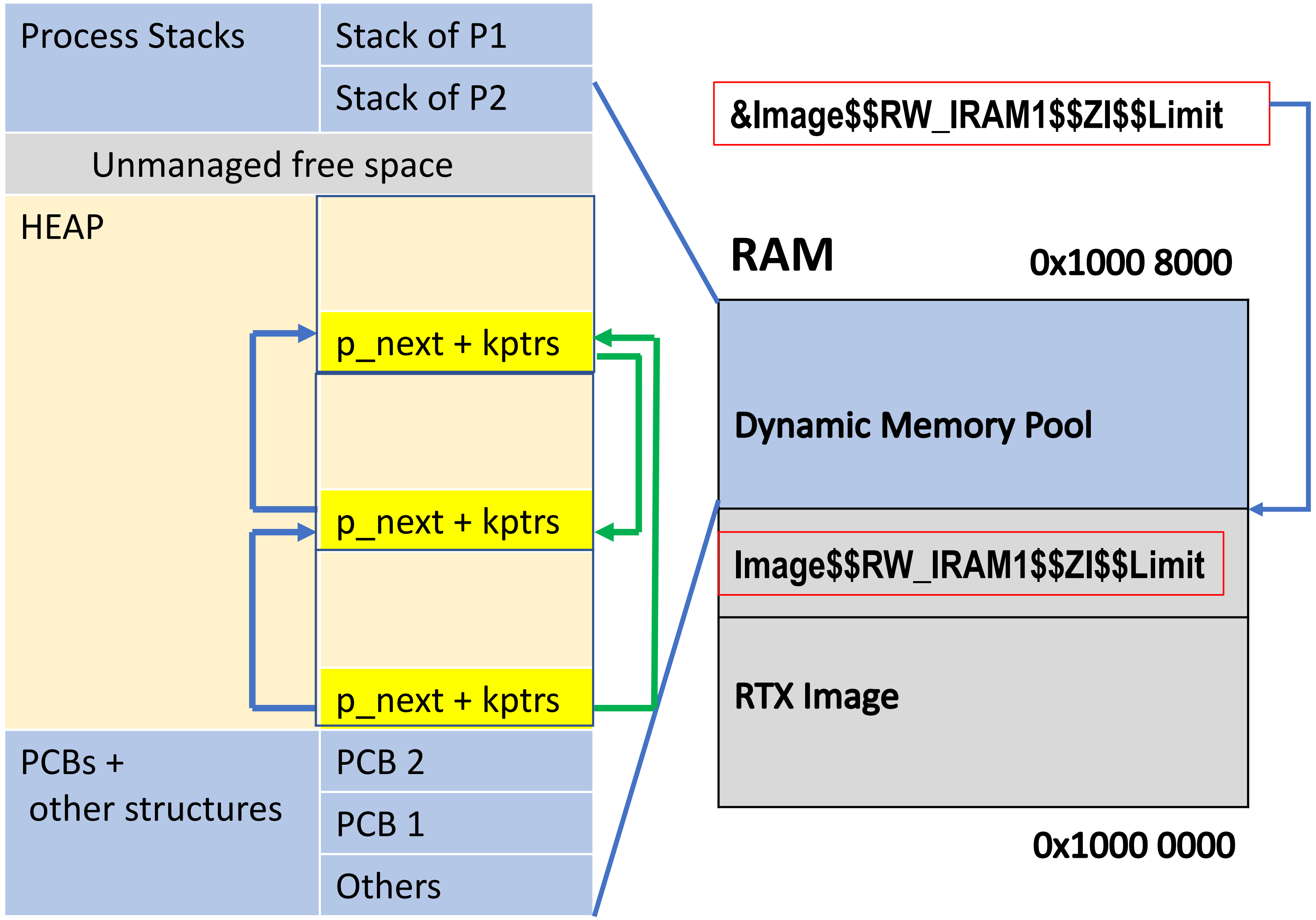
Linker defined symbol Image$$RW_IRAM1$$ZI$$Limit

```
extern unsigned int Image$$RW_IRAM1$$ZI$$Limit;

unsigned int free_mem = (unsigned int) &Image$$RW_IRAM1$$ZI$$Limit;
```

Copyright by Yiqing Irene Huang 2022. All rights reserved

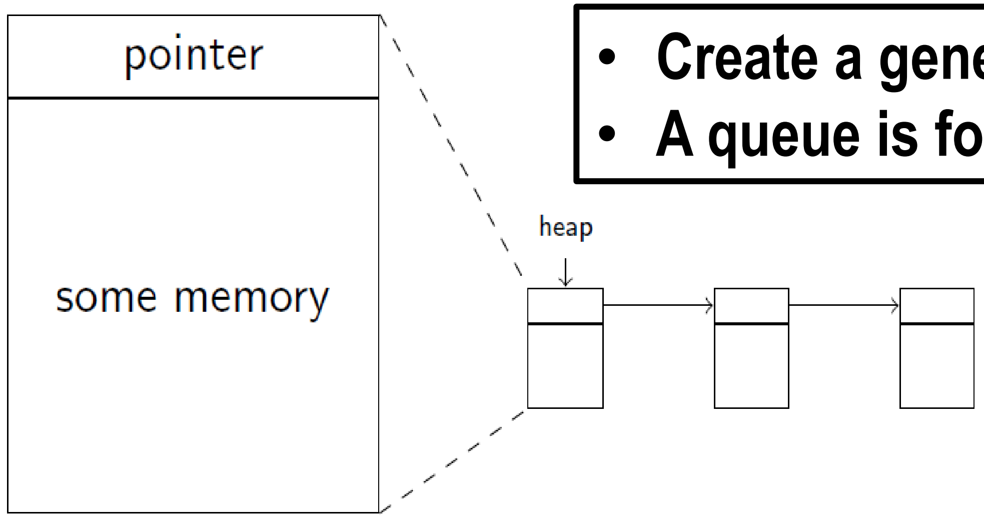
Fixed Size Memory Pool



Copyright by Yiqing Irene Huang 2022. All rights reserved

Memory Block Data Structure

- Linked list implementation of a queue



- **Create a generic purpose queue utility**
- **A queue is for managing a resource**

- Other implementations
- The caller turns the memory block to the proper data structure by **casting**.

Copyright by Yiqing Irene Huang 2022. All rights reserved

Memory Block Data Structure

- A generic memory block data structure

```
struct mem_blk {
        unsigned int mem_blk *next;
        /* other kernel pointers */
};
```

- No memory region variable(s) in the structure?
  - The size of each memory block and the number of memory blocks are configured with global constants at compile time (see common.h)
  - The list of free blocks is created during initialization phase.

Copyright by Yiqing Irene Huang 2022. All rights reserved

Memory Allocation

- A **non-blocking** memory allocator

```
void *k_request_memory_block_nb() {
        atomic(on)
        search for a free memory block in the pool;
        if (no memory block is available)
                return NULL;
        else
                update the data structure;
                return a pointer to the memory block;
        atomic(off)
}
```

- You will need this one for the i-processes in P2

Copyright by Yiqing Irene Huang 2022. All rights reserved

Memory Allocation Cont'd

- A **blocking** memory allocator

```
void * k_request_memory_block() {
       atomic(on)
       ptr = k_request_memory_block_nb();
       while (ptr is null) {
              put the PCB on blocked_memory_q
              set the PCB state to BLOCKED_ON_RESOURCE
              scheduler()
              process_switch()
              update ptr accordingly
       }
       atomic(off)
       return ptr;
}
```

Copyright by Yiqing Irene Huang 2022. All rights reserved

Memory Deallocator

- Note this may cause preemption to happen.

```
int *k_release_memory_block(void *mem_blk) {
       atomic(on)
       if (mem_blk is invalid)
              return ERROR_CODE
       if (blocked on memory resource q is empty) {
              put the mem_blk to free_memory queue/list
       else
              dequeue a blocked-on-memory PCB
              handle_process_ready(PCB)
              assign the mem_blk to the PCB
              scheduler()
              process_switch()
       }
       atomic(off)
       return SUCCESS_CODE
}
```

Copyright by Yiqing Irene Huang 2022. All rights reserved

Initialization

- What operations need to be carried out at start-up?
- Initialize all hardware, incl.
  - Board system Initialization
  - Memory mapping (memory allocation for mem blocks and stacks…)
  - ~~Interrupts (hardware and software: vector table & traps)~~
  - ~~Serial port(s) and timer(s)~~
- Create all kernel data structures
  - ~~PCBs (status=ready), queues…~~
  - ~~Place PCBs into respective queues~~

Copyright by Yiqing Irene Huang 2022. All rights reserved

Pointer Arithmetic

- Operation is in the unit of the size of the object the pointer points to.

```
int *p = 0;
unsigned int n = sizeof(int);
p++;
```

- p will be 4 on LPC1768. NOT 1!

Copyright by Yiqing Irene Huang 2022. All rights reserved

The sizeof Operator

- The sizeof (struct padding) is not 7, it is 8.
- Compiler adds padding.

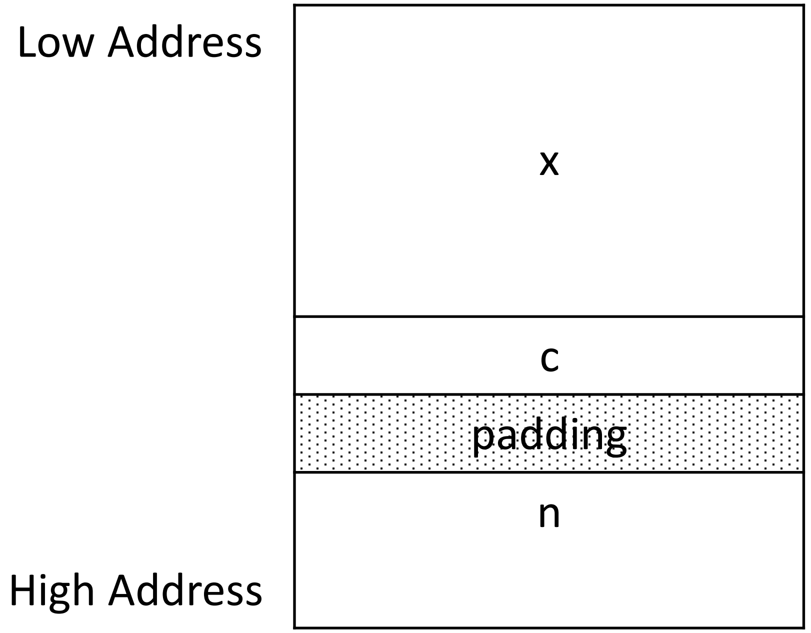
sizeof(struct padding)

$\neq$

sizeof(int) + sizeof(char) + sizeof (short)

- Little Endian

```
struct padding {
    int      x;
    char     c;
    short    n;
};
```



Copyright by Yiqing Irene Huang 2022. All rights reserved

Summary

- Study the starter code
- Create your generic linked list queue
- Create non-blocking memory allocator/deallocator

Copyright by Yiqing Irene Huang 2022. All rights reserved

References

1. Yiu, Joseph, *The Definite Guide to the ARM Cortex-M3*, 2009
2. *ARM Compilation Tools Version 5.0 Developer Guide*
3. *ARM Software Development Toolkit Version 2.50 Reference Guide*
4. *LPC17xx User's Manual*
5. *Software Interface Standard for Arm Cortex-based Microcontrollers, CMSIS Version 5.7.0*

Copyright by Yiqing Irene Huang 2022. All rights reserved

UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING

Thank you!

Electrical and Computer Engineering Department

Copyright by Yiqing Irene Huang 2022. All rights reserved