Tutorial 6.2

Separate Compilation

Lei Wang

lei.wang2@ucalgary.ca

UNIVERSITY OF CALGARY

Separate Compilation

Souce code (.c/.s) → Relocatable Object Code(.o)

Souce code (.c/.s) → Relocatable Object Code(.o)

…

link → executable file

first.s

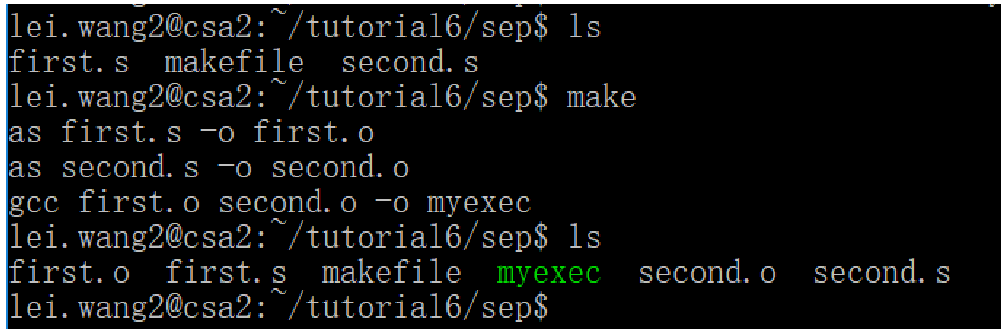
```
        .balign 4
        .global main
main:   stp     x29, x30, [sp, -16]!
        mov     x29, sp

        adrp    x19, a_m
        add     x19, x19, :lo12:a_m

        ldr     w0, [x19]
        bl      myfunc

        ldp     x29, x30, [sp], 16
        ret
```

second.s

```
        .data
        .global a_m
a_m:    .word   44

        .text
        .balign 4
        .global myfunc
myfunc: stp     x29, x30, [sp, -16]!
        mov     x29, sp

        sub     w0, w0, 1

        ldp     x29, x30, [sp], 16
        ret
```

makefile

```
all:
        as first.s -o first.o
        as second.s -o second.o
        gcc first.o second.o -o myexec
```



```
lei.wang2@csa2:~/tutorial6/sep$ ls
first.s  makefile  second.s
lei.wang2@csa2:~/tutorial6/sep$ make
as first.s -o first.o
as second.s -o second.o
gcc first.o second.o -o myexec
lei.wang2@csa2:~/tutorial6/sep$ ls
first.o  first.s  makefile  myexec  second.o  second.s
lei.wang2@csa2:~/tutorial6/sep$
```

C code call functions written in assembly

main.c

```c
#include <stdio.h>
int sum(int, int); // function prototype

int main()
{
    int i = 5, j = 10, result;

    result = sum(i, j);
    printf("result = %d\n", result);
    return 0;
}
```

sum.s

```
        .balign 4
        .global sum
sum:    stp x29, x30, [sp, -16]!
        mov x29, sp

        add w0, w0, w1

        ldp x29, x30, [sp], 16
        ret
```

makefile

```
all:
        gcc -c mymain.c
        as sum.s -o sum.o
        gcc mymain.o sum.o -o myprog
```

Separete compilation in assignment 4

a5aMain.c

```c
int main()
{
/* … … … code omitted here*/
switch (operation) {
case 1:
          enqueue(value);
          break;
case 2:
          value = dequeue();
          if (value != -1)
          printf("\nDequeued value is %d\n", value);
          break;
case 3:
          display();
          break;
case 4:
          printf("\nTerminating program\n");
          exit(0);
default:
          printf("\nInvalid option! Try again.\n");
          break;
}
/* … … … code omitted here */
return 0;

}
```

a5a.asm

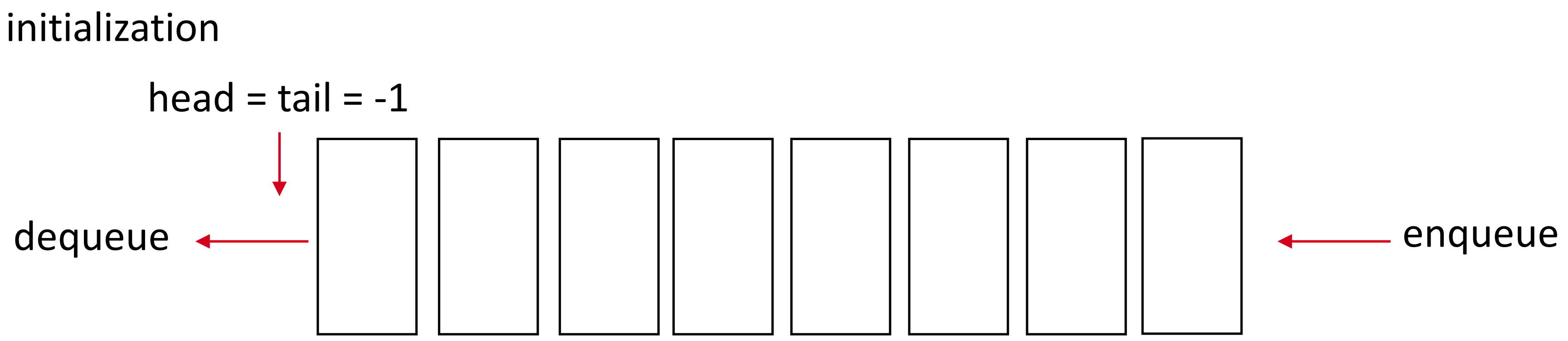
```
/*-------------------
macros and equates
---------------------*/
/*------------------------------
external variables(.data/.bss)
(with or without .global)
-----------------------------*/
/*------------------------
format strings
-------------------------*/

//equates for stack variables in enqueue
.balign 4
.global enqueue
/*------------------------
implementation of enqueue()
*/

//equates for stack variables in dequeue
.balign 4
.global dequeue
/*------------------------
implementation of dequeue()
*/

//equates for stack variables in display
.balign 4
.global display
/*------------------------
implementation of display()
*/
```

makefile

```
all:
        m4 a5a.asm > a5a.s
        as a5a.s -o a5a.o
        gcc -c a5aMain.c
        gcc a5aMain.o a5a.o -o a5a
```

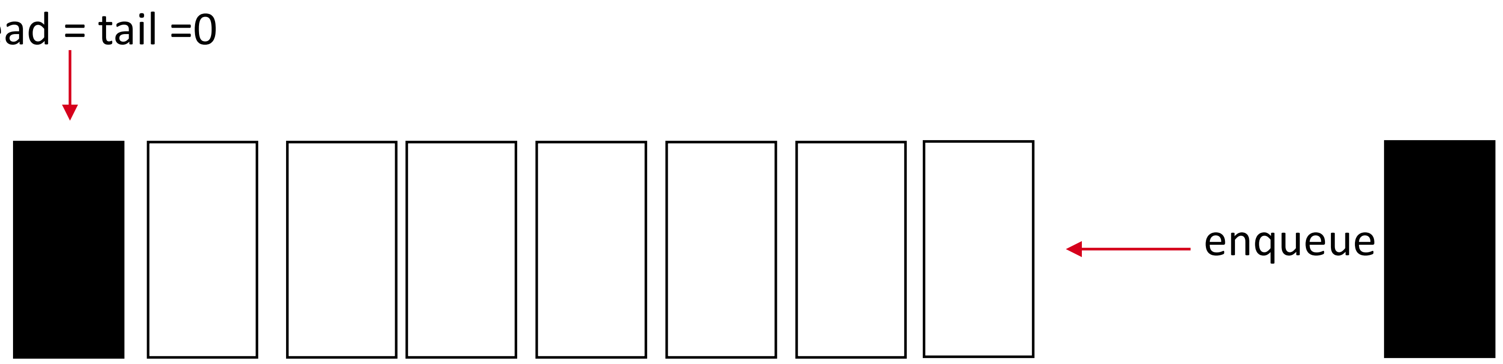
Assignment4 — FIFO queue data structure

- QUEUESIZE = 8
- implemented using array
- *head* and *tail* are the index of the first element and the last element
- when there is only one element in the queue: $head = tail$

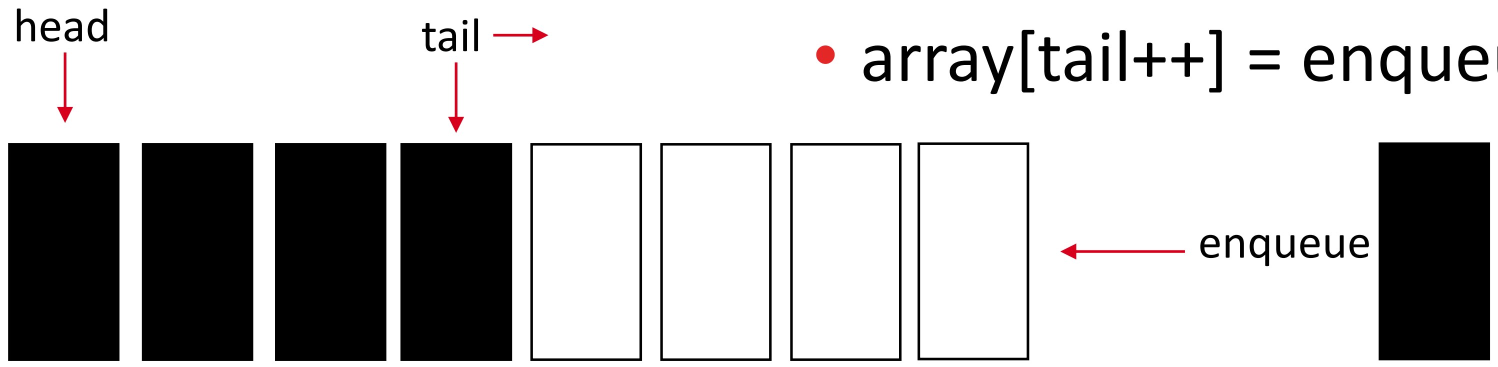


Enqueue

- when the first element is enqueued, set *head* and *tail* both to 0

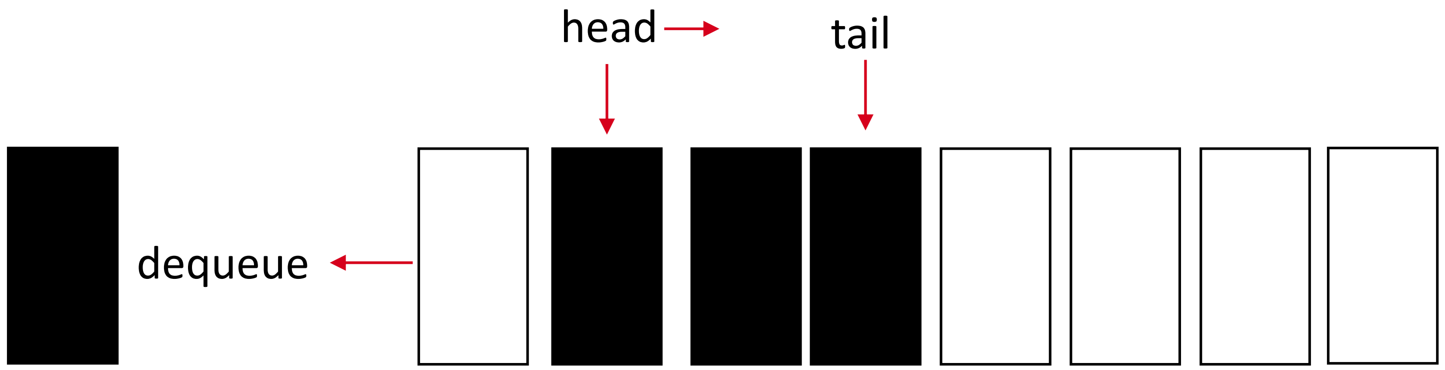


- when an element is enqueued, set *head* unchanged and *tail*++
  - array[tail++] = enqueued_value



Dequeue

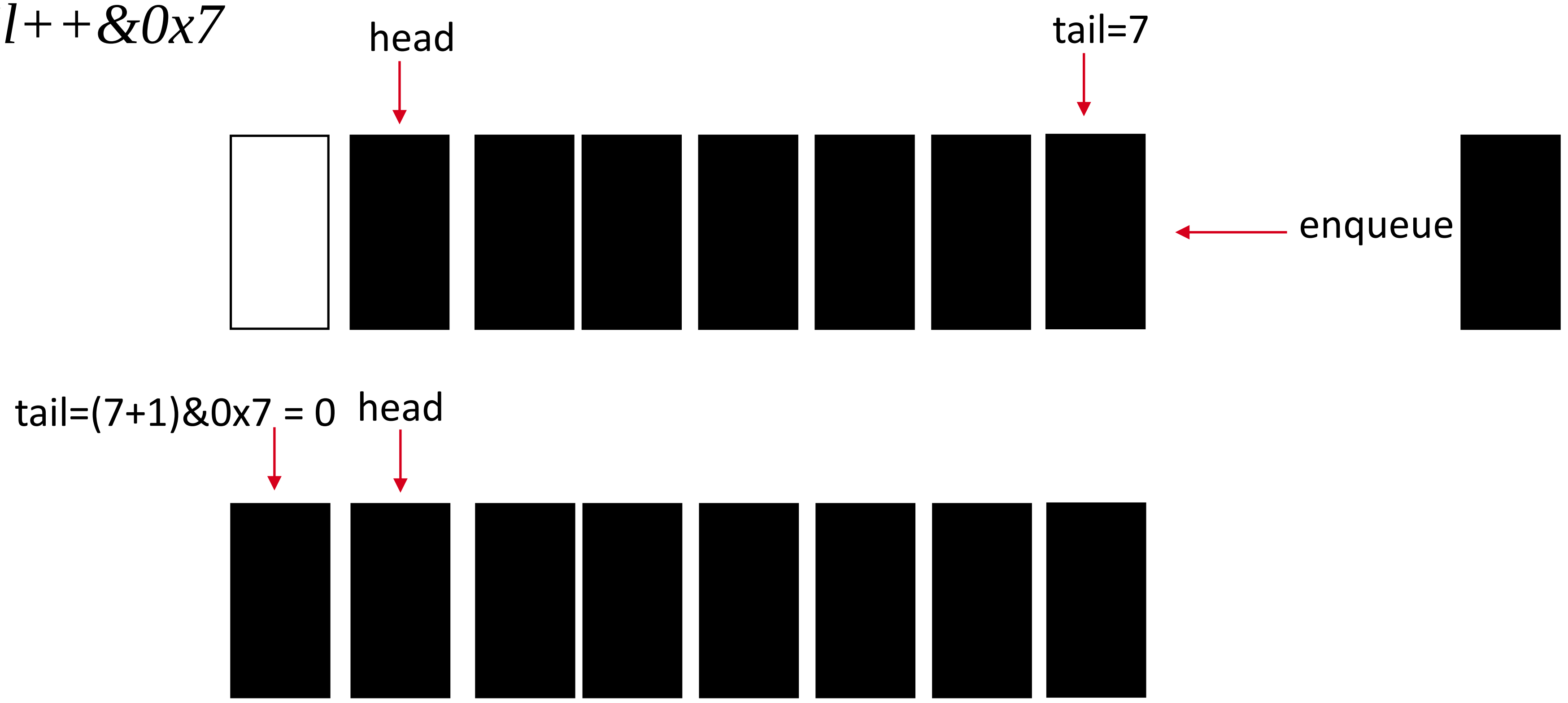
- when a element is enqueued, set *head*++ and *tail* unchanged



- dequeued_value = array[head]

Recycling the array

- when incrementing *head* and *tail* and array is not full, set them to *head++&0x7*, *tail++&0x7*



- &0x7 is equal to %8, is to get the reminder of the division by 8
- when (tail+1)&0x7 == head (in enqueue), the queue is full
- when tail == head (in dequeue), the queue is empty

Assignment4: external variables

- queue
  - array of 8 ints
  - size: 8*4
  - uninitialized
- head
  - index of the first element
  - size: 4 (word)
  - initialized with -1
- tail
  - index of the last element
  - size: 4 (word)
  - initialized with -1