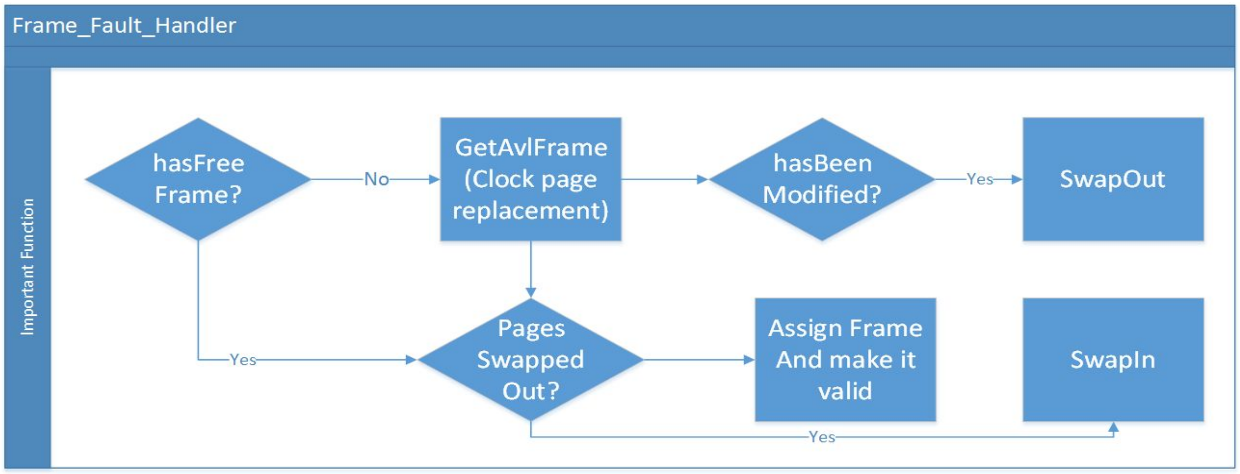
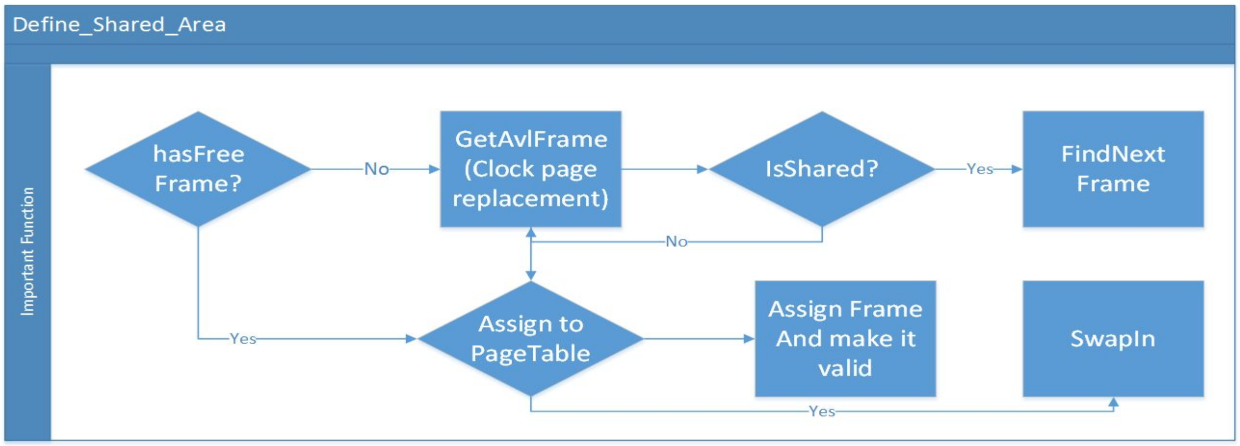
Architectural Document

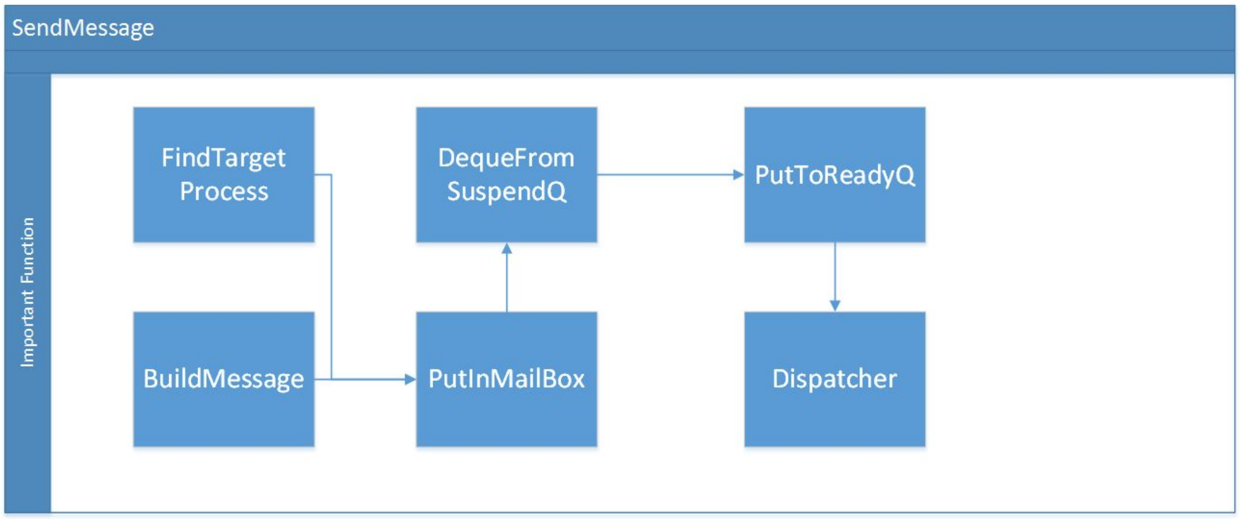
A . what is included

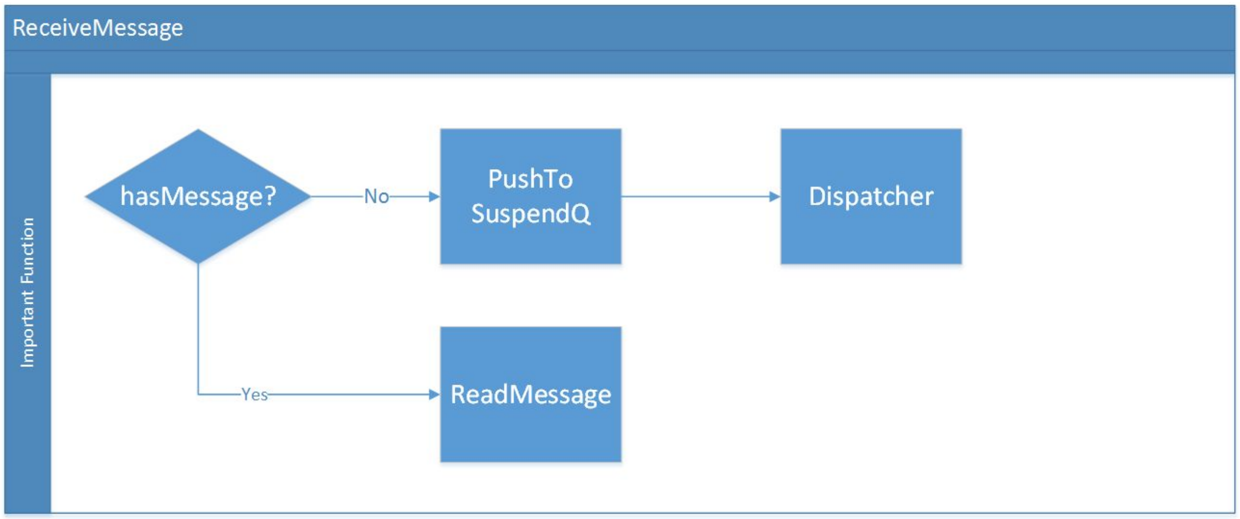
What have done	Details
Fault Handler could assign free frame or find victim.	By using CLOCK algorithm to find victim frame.
Fault Handler won't touch shared frame.	By using "pin" mark to tell which one is shared which is not.
Finish message passing between processes.	By building mailbox to connect two process. And adding a suspend queue to postpone the receive.
Interaction between process by using shared memory	By assigning frame to maintain a shared area.

B. High Level Design









C. Design justification

There are several advantages for this design

- This is a simple design, each component works independently. When page fault occurred, it will find a free frame, and also, to avoid the switch context problem, we can cache the swap area, and after finishing deal with page fault, flush it to disk.
- To the shared area part, since each tag has its own frames, so I build a table to save each tag's frame, and then assign them to corresponding page table. And another method of sending and receiving message. Since there is a possibility that no messages send to. We need to suspend it, so I push the pcb to the suspend q, until master finish sending message.

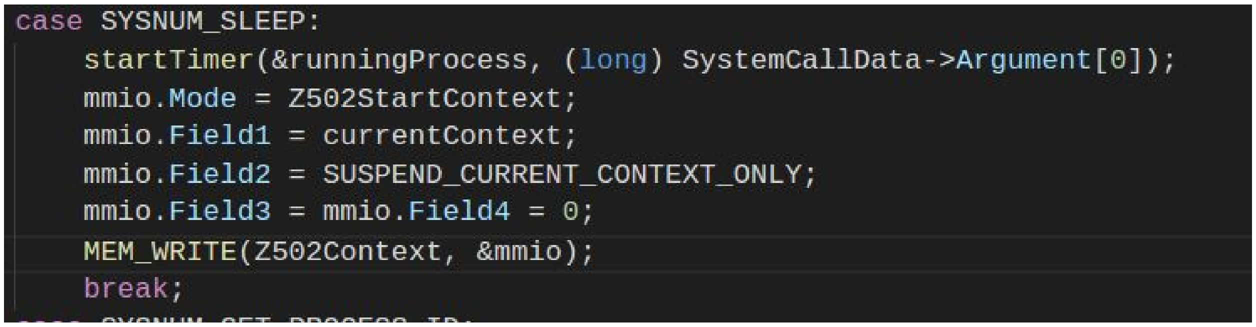
D. Unique part

- Each process has its own mail-box, which is convenient to send and receive message, and the mail-box is saved in process control block, so it is very easy to find process by using process id in the table.
- Make frame fault handler work in automatically in case of when doing swap in or out, other process get page fault and the information is not updated so that the swapped process get wrong result.

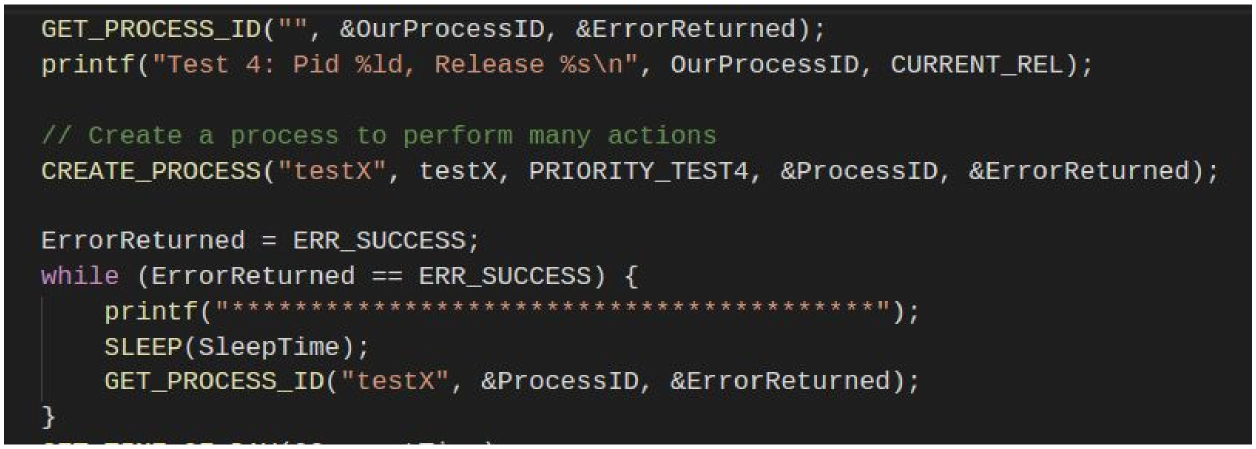
Small. bug

I tried to develop multiprocessor version, however, I developed in linux system, and it seems that start context suspend doesn't work, since after I suspend it, the simulation continue to do things for its test.

Here is screenshot:
I call suspend current context only for test4 in SLEEP system call. However the while loop continue work, and it seems that didn't suspend this current context.



```
case SYSNUM_SLEEP:
    startTimer(&runningProcess, (long) SystemCallData->Argument[0]);
    mmio.Mode = Z502StartContext;
    mmio.Field1 = currentContext;
    mmio.Field2 = SUSPEND_CURRENT_CONTEXT_ONLY;
    mmio.Field3 = mmio.Field4 = 0;
    MEM_WRITE(Z502Context, &mmio);
    break;
```



```
  GET_PROCESS_ID("", &OurProcessID, &ErrorReturned);
  printf("Test 4: Pid %ld, Release %s\n", OurProcessID, CURRENT_REL);

  // Create a process to perform many actions
  CREATE_PROCESS("testX", testX, PRIORITY_TEST4, &ProcessID, &ErrorReturned);

  ErrorReturned = ERR_SUCCESS;
  while (ErrorReturned == ERR_SUCCESS) {
      printf("****************************************");
      SLEEP(SleepTime);
      GET_PROCESS_ID("testX", &ProcessID, &ErrorReturned);
  }
```