1. IPC DIRECTIVES

This section describes the inter-process communications directives, alias IPC. IPC consists of three seperate and dinstinct operations. They are:

- Message queues
- Seamphores
- Shared memory

The following sections will describe these in more detail.

1.1 Message Queues

Message queues are used to send messages from one process to another process. The process that is receiving messages must allocate a queue before another process can write to this queue. If a process is a transmitting process it is not necessary to allocate a queue. However if it expects messages than it must allocate a queue also. After a queue is allocated, it can be written to. The allocating process is the only process that can read from this queue. Other process that try and read from a queue that and they are not the owner, will receive an error message. There are three functions that a process can perform for messages queues:

— allocate a queue

— read a queue

— write a queue

1.1.1 Allocating a message queue

A process that expects to receive messages must allocate a queue before other process can write to it. This is accomplished with the "msgget" operating system call. The calling process must initialize its registers to the following:

Operating System Interface

Register Usage:

— R0 - Key

— R1 - Flags

Parameters:

Key

The key is a 16 bit binary number and must be unique in the system. It is assumed by the operating system that cooperating process known each other keys. The operating system will allocate a message queue with the requested id, if that id does not exist. After the allocation is performed the operating system returns the queue id. This is used by this process in subsequent calls to the read system call.

Flags

Flags indicate to the operating system weather you are creating this queue or requesting the queue id to write to another queue.

Output:

— SUCCESS - R0 = Queue id

— FAILURE R0 – Negative error number.

1.1.2 Read queue

After a process has allocated a message queue it can read from this queue. If a read is performed, then the process must initialize its registers per the following:

Operating System Interface

Register Usage:

- R0 - Queue id
- R1 - Destination buffer address
- R2 - NA
- R3 - Flags

Parameters:

Queue Id

The queue id is the number returned from the operating system. If this number is not valid, then the operating system will return an error.

Destination Address

The destination address is a users buffer. This must be a word orenated address, and the first word must indicate the size of the buffer. This is a positive number. If the buffer is not large enough for the current message or if the number is negative, than the operating system will return an error.

Flags

The flag field determines what action the operating system can take when no messages are in the buffer. This must be set to "BLOCK" in version 1.0.

Output:

- SUCCESS - R0 = Zero
- FAILURE - R0 = Negative error number

1.1.3 Write Queue

After a process has attached itself to a queue, then it can write to another queue. This is accomplished by initialization the following registers:

Operating System Interface

Register Usage:

- R0 - Queue id
- R1 - Source Buffer
- R2 - NA
- R3 - NA

Parameters

Queue Id

The queue id is the number return from the operating system. If this number is not valid, then the operating system will return an error.

Source Buffer Address

The users buffer is an word orenatied address with the first word initialized to the size of the message in the buffer. NOTE: this is not the size of the buffer, but the size of the message.

Output:

- **SUCCESS - R0 = Zero**
- **FAILURE - R0 = Negative error number**

1.2 Process Termination

When a process terminates. the operating system will free up the queue allocated an flush all remaining messages in the queue. The queue id is now invalid. after this is accomplished. If another process tries to write to a queue that was removed. an error will be returned.