MEMOREX 7100

OPSYS1 EMULATION PACKAGE

R. CHUEH

R. HOEHNLE

7/19/72

CONTENTS

1.0 INTRODUCTION

This section will outline the 7100 Emulation Package. The Emulation Package has been designed so that OPSYS1 can be executed on the 7100 CPU with minimum modification. The OPSYS1 has been designed for the 7300 CPU which has a time-slicing multi-processor structure. The emulation of the multi-processor states is the main concern of this section.

1.1 THE 7300 CPU

The 7300 CPU is organized into eight (8) semi-independent processor-states. Each processor-state is assigned a particular processing job and is competing with each other for shared CPU sources and central memory access. The CPU is time sliced into time slots of major cycles, (0.9 μs or 1 μs). At the end of the cycle a new processor state is scheduled for the next cycle. The active processor states are scheduled in a "round-robin" algorithm supplemented by dynamic priority logics. Switching from one processor-state to another is hardware controlled and is overlapped with the normal instruction execution. For detailed description of 7300 CPU organization, please refer to its Production Description.

2.0 Priority Structures of the 7100 CPU

7100 has three levels of priority structures. The Central Memory assigns fixed priorities to its accessing ports from the highest refresh port to the lowest CPU port. Within the CPU, fixed priorities are also assigned to its processor-states associated with columns of the register-file. The higher priority processor can "interrupt" any of the lower priority processors at the "break" points. Each I/O adaptor is assigned to a processor-state. The majority of the non-time-dependent

data processing jobs are scheduled on the processor state F, the lowest priority one. To emulate OPSYS/I within the processor-state F, software will simulate the 8-state assignments of MRX/50 for non-critical-time-dependent jobs.

2.1 Memory Accessing Priority

Accessing priorities to the Central Memory have been fixed in the following order:

Refresh

DMA 1 (Assigned to DISC data transfer)

DMA 2 (Reserved for Selector Channel data)

DMA 3 (Reserved)

(SPARE)

(SPARE)

(SPARE)

CPU

Switching from one port to another:

Switching Quantum = Memory Cycle = 1.2 us

Switching (Transition) Time = 0

2.2 CPU Processor-State Assignments

The Register-file in CPU is partitioned into columns with each column associated to a processor-state. At any instant, only one of the processor-states is active.

The P-register identifies the current active processor-state and so the associated column of the register-file. Normal addressing to the register-file is restricted to the column of the active processor-state. (Address to other columns can be specified through the use of the X-register).

Fig. 1 depicts the organization of the Register File. There are 16 columns but only 9 columns have registers, the columns from 8 to E are empty. In the first 8 columns (processor-state 0-7), only half of the 16 registers per column are presented in the register file, the rest are located in the various I/O adaptors. These "Virtual I/O Registers" can be addressed as if they were part of the register-file, the meaning and interpretation of a fetch or store to these "virtual registers", however, are part of the particular I/O adaptor organization.

The 9 processor-states are assigned as:

Processor State	Assignment
0	DISC Commands
1	Selector Channel Commands
2	Communication Adaptor
3	Printer Adaptor
4	Card Reader Adaptor
5	MFCU
6	Console
7	Timer
F	General Data PRocessing

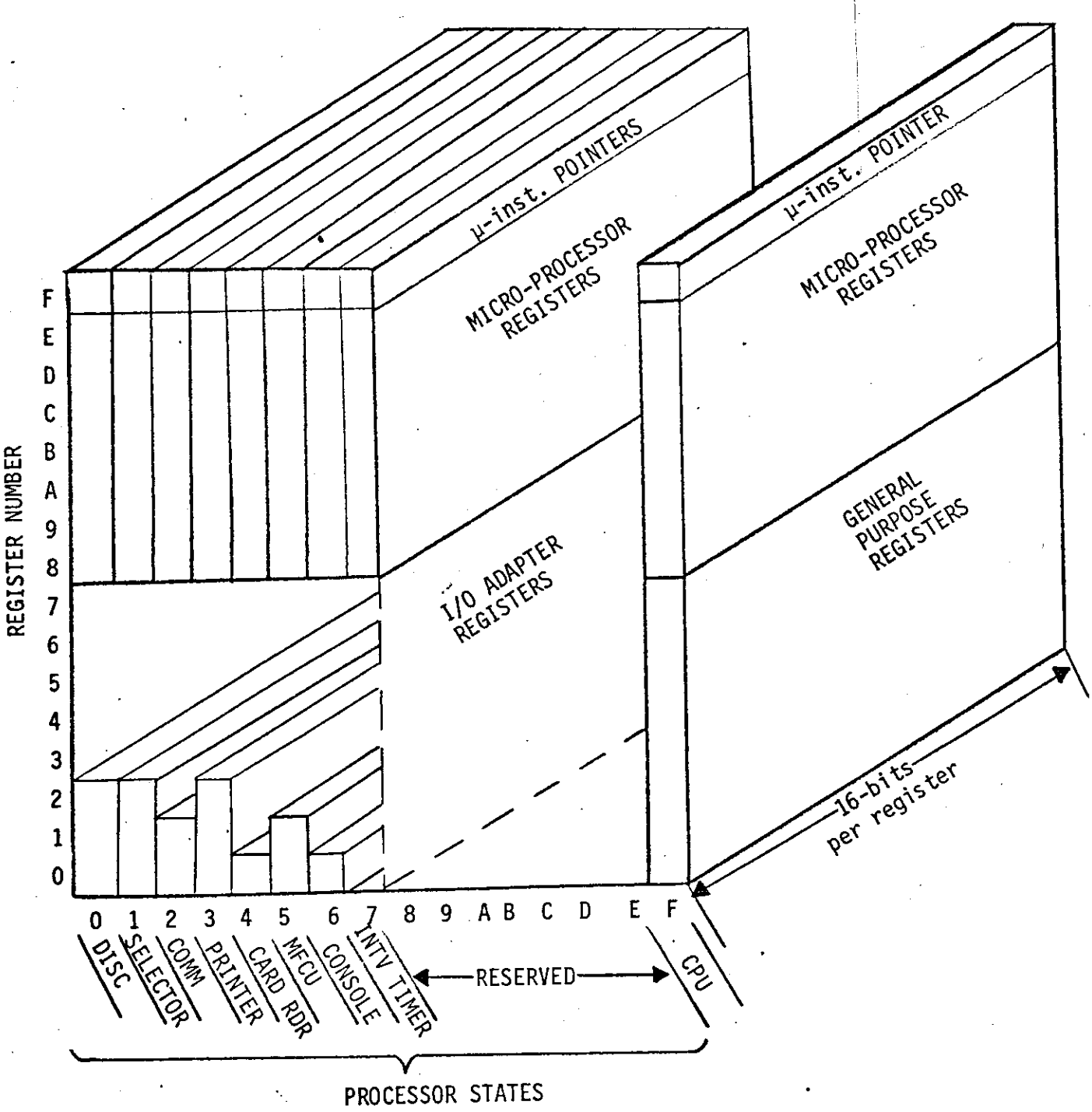

FIGURE .I

7100 SYSTEM REGISTER FILE

2.3 CPU Processor-State Switching

The process of interrupt and switch to higher priority procedures in 7100 CPU is accomplished with a priority network and a four-instruction BREAK subroutine. A processor-state can request service by raising the Service Request Line of the processor (usually by hardware logic in response to an external condition). The priority network will select the highest priority processor-state (lowest in Hex number) which has the Service Request Line raised. The select processor-state is encoded into a four-bit number which can be loaded into P-register when a "CST" micro-instruction is issued. There is also a comparison netowrk that compares the four-bit value with the current P-register and the output of the comparison can be tested by a conditional branch micro-instruction.

The micro-routine to test and execute a higher level processor state interrupt is:

```
BREAK:   Branch to Next if no higher request; else Branch to
         Switch, save return in R(F).

NEXT:

SWITCH:  Gate "CST" to P

         Branch to location specified by R(F)
```

When a processor-state finishes its current processing, it will have its Service Request Line reset and executes the following micro-instruction:

FINI: Branch to Switch, Save Return in R(F).

Switching from one processor-state to another:

Switching Quantum = Maximum Time Span between two BREAKS

Switching (Transition) Time:

The BREAK (including SWITCH) = 5 x 0.4 = 2 μs (taken)
= 0.4 μs (not taken)

The FINI (including SWITCH) = 4 x 0.4 = 1.6 μs

The maximum Time Span between two BREAKS is a critical parameter and is calculated to be 30 μs

2.4 The Processes in State F

The lowest priority processor-state (F) has 16 registers and is used to process all non-time dependent jobs. Within the processor-state F, processes are organized to emulate the 8-processor-state of 7300 CPU. The scheduling of the 8 simulation processes is of major concern in the following section. Since all time-dependent jobs has been "farmed-out" to either DMA channels or high priority processor-states (0-7) there is no critical timing requirement on the scheduling of the processes. It has, however, to be logically correct in emulating 7300 processor-states, to avoid possible dead-lock situation and to improve overall system performance.

3.0 The Busy and Active Bits of 7300 CPU

Two flip-flops (the busy and active bits) are provided for each processor-state in 7300 CPU. These bits are used to indicate the state of the processor and the Busy bit is also used for requesting CPU cycle (time-slot). When a processor has its Busy bit reset, it is either not active or is waiting for an external signal to wake it up. To distinguish the "wait" state from the "in active" state, an Active bit is provided.

The Busy and Active bits are also used as attention signals for sending messages from one processor to another. When a processor sends a message to another, it attaches the message into a queue in the central memory and sends an attention signal to wake up the receiving processor-state. If the receiving processor-state is already running, the attention signal will be ignored; if it is waiting for some specific completion signal, it should not be disturbed (do not wake-up by the signal). The logic can be implemented in two ways: one is to have the attention signal masked by the receiving processor, the other is restraining the sending processor from issuing the attention signal. The MRX/40 and 50 systems adapt the latter discipline. For example, when the executive Processor (4) issues a job to an I/O Processor, it will put the job description into a work queue and set both Busy and Active bits on the I/O Processor if it is not already Active. If the receiving

processor is already Active, it is either running (Busy) or is waiting for an I/O completion signal. In either situation, the Executive Processor then will not disturb the Busy or Active bits of the receiving processor. The message attached to the work queue will only be processed after the I/O processor has serviced the I/O completion signal or the current running job.

When a processor-state is serving more than one device (e.g. the Communication Processor in the 7300 CPU controls up to 16 lines), the functions of the Busy/Action bits have to be somewhat modified. The Busy bit is still used to request CPU cycles. To display whether each device controller is waiting for a completion signal, an (Active) bit is needed for each device (line). Since the processor is now controlling more than one device, waiting completion signal from one device (having the particular Active bit set) should not block the processor from processing jobs issued for other devices. The "non-disturb" discipline of single device processor has to be modified into a multiplexing organization. In addition, the completion signals have to be held high indefinitely since they may not be recognized immediately.

4.0 Organization of the Emulation Package

Fig. 2 outlines the organization of the emulation package for execution OPSYS/I in 7100 CPU. Basic to the emulation package and not shown in Fig. 2 is a machine language emulator that emulates all non-privileged MRX/50 instructions.

In OPSYS/I, every processing job is assigned to one of the 8 processors. Programs executed by processors 4 to 7 are exclusively machine language subroutines. Programs executed by the four I/O processors (0-3) include both micro-command and machine language routines. The general approach of the emulator is to create an 8-processor organization within the 7100 CPU state F so that most of the machine language programs can be executed with minimum or no modification. The micro-command I/O routines that handle I/O command initializations and terminations as well as low speed data transfers are rewritten in the 7100 micro-commands and to be executed in dedicated 7100 CPU states (0-7). In terms of system/360, state 0 to 7 correspond to eight "I/O channels" and state F corresponds to "CPU".

The disc and selector channel data movements, processed by short micro-command loops in 7300 CPU, are transfered to be handled by hardwired logics in DMA channel #1 and #2.

As we have discussed in the last section, the states of the 7300 processors are indicated by their Busy/Active bits and they are scheduled accordingly. The Busy bits are still used to indicate that a simulating process is in a ready state. The allocation of the CPU resources to the process, however, is quite different. Instead of time slicing on every major cycle, processes are usually allowed to execute (in state F) to completion. Changing from one process to another includes the lengthy operation of swapping (state F) CPU registers. The exact processes scheduling algorithm which involves the Tie-Breaker Register and Timer is detailed in the last section.

To activate an I/O state (0 to 7), the responsible CPU process loads its micro-command pointer (register F of the I/O State) and sets its Service Request Line. All states have their service request lines set are competing for execution at next program BREAK point. When a process-state finishes its current processing, it will have its Service Request Line reset and signals the responsible process (via its Busy/Active bits) before it issues a BREAK.

Integrated with each I/O state is an I/O adaptor for controlling a particular device. 8 of the 16 register addresses of the state are reserved for addressing adaptor registers. For disc and selector channel adaptors, they also include registers of the DMA's and thus provide communication and control of DMA operations.

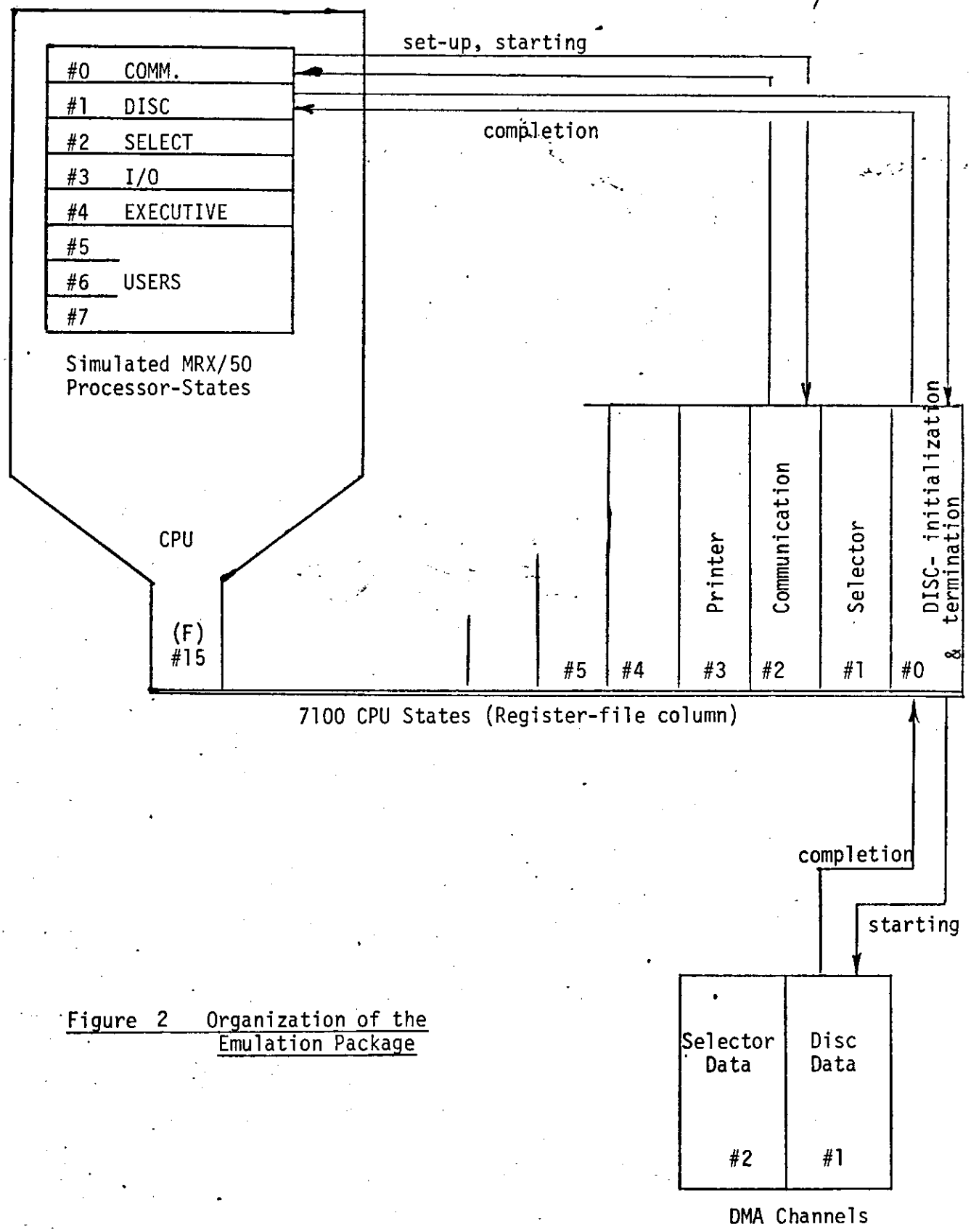

Figure 2 Organization of the Emulation Package

5.0 The Tie-Breaker Register and Process Scheduling

The Tie-Breaker Register is a hardware facility provided by the 7300 CPU to ease the implementation of synchronization of processors. In OPSYS/I, at the entry of a work queue a processor tests-and-sets a bit designated for the work queue in the Tie-Breaker Register. If the bit has already been set, the processor will loop and test again unitl the bit is reset.

The 7100 is a single processor system. Switching from one processor-state to another can only occur at program designated BREAK points. Therefore, the processors synchronization logic of test-and-set can be implemented with a simple fetch-branch-and-store macro provided no process switching is allowed inside the macro.

To emulate OPSYS/I in the 7100, the convention of simply looping if the Tie-Breaker Register is found set has to be somewhat modified. Because the processors in the 7300 CPU are scheduled cycle by cycle on "round-robin" basis, there is no danger of getting into "dead-lock" situation. The corresponding emulation processes are scheduled to run to completion in the 7100. Simple looping may create a "dead-lock" situation. The following conventions are adopted for the emulation package:

(1) A Process becomes ready when its Busy bit is set. A process becomes not ready when its Busy bit is reset. A process becomes blocked when it tests the Tie-Breaker Register and found the bit set. A process becomes not blocked when the bit in the Tie-Breaker Register is reset.

(2) Among the ready and not blocked processes, CPU is allocated to the highest priority process. The priority is fixed according to the process number (0 to 7).

(3) An allocated process is normally scheduled to run to its completion (in Processor-State F)* except at the occurance of the following conditions:

(a) Timer Expiration

(b) Manipulation of Busy bits

(c) Clear Tie-Breaker Register

(d) Test Tie-Breaker Register and found the bit set

(4) The occurance of condition (a), (b), or (c) will result to a process switching to a higher priority process which is ready and not blocked. If the current process is the highest one, no process switching will result.

(5) The occurance of condition (d) will always result to a process switching to a higher priority process which is ready and not blocked.

---

* (Remark) All processes (programs) executing in Processor-State F have to provide BREAKS for high-priority I/O Processor-State at least every 30 μs.