論文 76-13-2-6

NOVA 에뮤레이터의 시뮤레이숀에 관한 硏究

(A Study on Simulation of NOVA Emulator)

宋 榮 宰*

(Song, Young-Jae)

要 約

本論文은 Minicomputer를 LSI화 할때의 문제점을 명확히 하고, 그 解決策을 검토한 다음 앞에서 設計한[1] NOVA Emulator의 性能을 評價하기 위한 Simulation을 行하였다. 成果로서는 Microprocessor에 의한 하드웨어設計의 문제점과 몇가지 課題등이 명확히 되었다.

Abstract

The purpose of this thesis is to make clear the problems which would arise from the process of making the Minicomputer by the employment of LSI, and to examine a solution to the problems. This Simulation to do for value performance of NOVA Emulator which designed before this[1].

As a result of this studies,

The problem of the hardware design by Microprocessor, the problem to be accompanied with application of LSI to the computer in the furture, etc. are mentioned definitely.

1. 序 論

컴퓨터의 高度化 경향에 따라서 일반적으로 컴퓨터의 設計는 매우 어렵고, 많은 시간과 노력을 필요로 하기 때문에 주어진 仕樣을 最良의 條件으로 만족시키기 위하여는 그 시스템을 Simulation하여 性能을 評價할 필요가 있다. 컴퓨터 시스템의 Simulation은System Performance의 分析 및 평가를 하기 위한 有力한 하나의 방법이다[2]. 이것은 대상이 되는 컴퓨터 시스템을 다른 컴퓨터에 의해서 그대로 模擬하는 방법이다.

컴퓨터 시스템의 設計에 관련한 시뮤레이숀에는 여러가지 방법이 있지만, 컴퓨터의 Hardware構成에 관한 것만을 限定한다면 다음 2가지의 커다란 연구과제가 있다[3]. 하나는 주어진 仕樣을 만족하게끔, 설계한 시스템의 論理回路에 관한 Simulation이고, 또 하나는 그 시스템 전체의 動作特性에 관한 Simulation이다.

本 硏究에서는 著者가 앞서 設計한[1] NOVA Emulator가 정확히 동작하는가를 확인하기 위함이다. 여기에서는 Target 計算機(NOVA Emulator)를 Host計算機(NOVA 1200)上에 Software로 표현해서 Target 계산기의 프로그램을 Host계산기上에 實行시키는 방법을 모색한다.

2. Simulator의 構成[4][5][6]

앞서 著者에 의해 NOVA 1200上에 設計한 NOVA Emulator를 사용하여 각종 Hardware의 Subroutine을 만들어 Control ROM의 命令을 Main Routine으로서 User의 프로그램을 실행시킨다. Subroutine으로서 필요로 하는 것은 MMI-6701의 하드웨어, PLA의 하드웨어, A/B Ports Select Multiplexer의 Routine이다. 또, 이밖에 Emulator에 사용한 Register등 으로서 Instruction Register, Memory Address Register, Carry Flag, Interrupt On Flag를 지정한다. Emulator에 있어서 입출력기구 및 命令語는 NOVA와 같기 때문에 Emulator의 프로그램이 NOVA의 입출력처리

*正會員, 慶熙大學校 電子工學科
Dept. of Electronic Engg.
Kyung Hee University
接受日字 1976年 5月 29日

를 실행하는 방식을 취한다. 假想 Main Memory는 Simulator에 격납시켜 있는 부분의 다음에 계속된다. User Program의 Main Memory에의 Load는 Loader로서 假想 Main Memory에 Load시키는 것처럼 조절한다.

3. Simulator의 設計

(i) NOVA Emulator와 NOVA의 Architecture對應

NOVA Emulator에 사용되고 있는 Hardware를 그림 1과 같이 NOVA의 Main Memory에 設定한다.

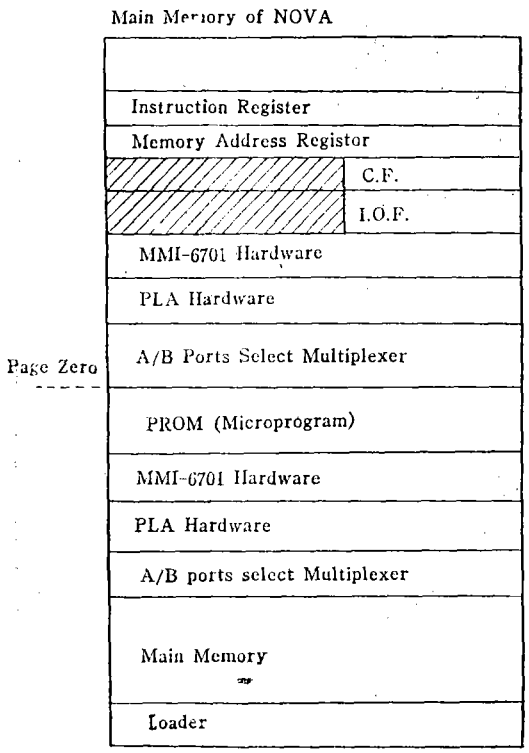

그림 1 시뮤레이터의 設定
Fig. 1 Design of Simulator

각각 Hardware의 Simulator는 Page Zero Routine과 Page Zero이외의 Routine을 組合해서 하나의 것을 구성한다.

(ii) 각종 Hardware의 Subroutine

각종 Subroutine에 대해서 Main Routine (Control ROM Routine)과의 관계를 표시하면서 記述한다.

A) MMI-6701 Hardware Routine

MMI-6701의 Hardware Routine을 NOVA의 Main Memory上에 그림 2와 같이 設定한다.

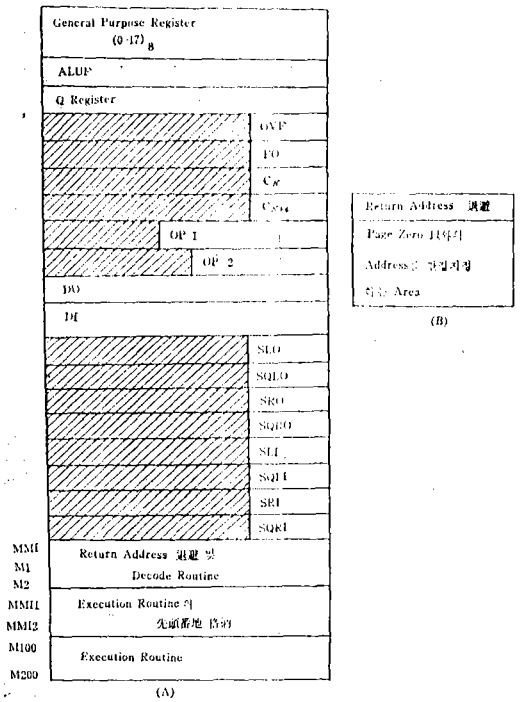

그림 2 MMI-6701의 設定
Fig. 2 Design of MMI-6701

그림 2의 (A)는 Page Zero 以外, (B)는 Page Zero이다.

Control ROM의 Routine에 있어서 MMI-6701이 實行에 사용되는 Data를 OP1, OP2, C_N, A Port, B Port에 格納해서 MMI番地에 Jump Subroutine한다.

MMI番地로 시작되는 Routine은 Control ROM Routine의 Return Address를 退避시키는 Routine이다.

MMI-6701의 명령은 ALU명령(OP1)과 Shift, Load, Data Out Control (OP 2)의 2종류의 명령이 있다. OP1과 OP2를 해독하는 Routine이 각각 M1, M2에 격납되어 있다. 또 OP1, OP2의 實行Routine은 각각 M 100, M200에 격납되어 있다. 그것의 실행Routine의 先頭番地가 MMI1, MMI2에 격납되어 있다. OP1과 OP2의 해독은 Displacement를 OP1, OP2의 내용, 각각의 Base Register의 내용을 MM1, MM2라고 하는 Index Address方式에 의해 실행 Routine 先頭番地의 내용에 分岐하는 간접Address方式을 취한다.

命令實行은 OP1, OP2의 순서로 行하여지기 때문에 OP1의 實行Routine의 最終Step에서 반드시 M2番地에 Jump한다. OP2의 실행 終了後는 MMI번지로 부터의 Routine으로 대피시켜 놓았던 Control ROM Routine

의 Return Address에 돌아간다. General Register의 지정에 있어서는 A Port, B Port의 내용을 Displacement로서, Base Register의 내용을 GR이라고 하는 Index Address방식을 취한다. 그밖의 레지스터 지정은 Page Zero를 사용해서 간접Address方式을 취한다.

B) PLA Hardware Routine

PLA Hardware Routine을 NOVA의 Main Memoy 上에 그림 3과 같이 設定한다.

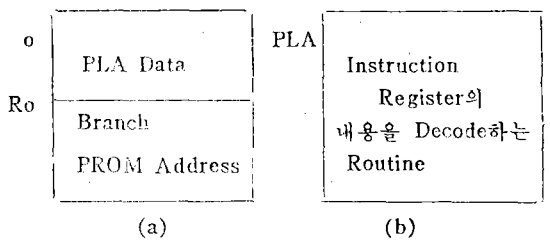

그림 3 PLA의 設定
Fig. 3 Design of PLA

그림 3의 (a)는 Page Zero (b)는 그 以外. Control ROM Routine에 있어서 해독에 필요한 PLA Data의 격납번지와 그때의 브란취先의 Control ROM Routine Address를 Index Register에 격납하여서 PLA번지에 짐프한다.

PLA번지로부터의 Routine은 먼저 User프로그람의 명령어 가운데 하나의 실행에 필요한 명령의 bit만을 남기고 나머지 bit는 모두 Zero로 한다. 다음에 이와 같이 하여 필요한 bit만이 남은 命令語와 Index Register에 격납시킨 Address의 PLA Data와의 差를 취해 Zero이면 Index Register에 격납시킨 Control ROM Routine의 Address에 브란취한다.

PLA Data의 어떤 語(Word)가 격납되어 있는 Address를 'P', 그 語의 bit pattern에 의하여 실행시켜야만 할 명령의 Control ROM Routine의 Address가 격납되어 있는 번지를 'R'이라고 한 경우, PLA Data가 격납되어 있는 先頭番地를 'Po', 브란취를 위하여Control ROM Routine의 번지가 격납되어 있는 선두번지를 'Ro'라고 하면 'P'와 'R'은 똑같은 Displacement 'D'로 표시 된다. 즉 P=Po+D, R=Ro+D이다.

C) A/B Ports 選擇 Multiplexer의 Routine. A/B Ports 선택 Multiplexer의 Routine을 Main Memory 上에 그림 4와 같이 設定한다. 그림 4 (a)는 Page Zero, (b)는 그 以外이다.

MUXA에는 IR의 bit 1~2, MUXB에는 bit 3~4가 들어간다. 이 Routine은 IR의 bit 1~2, 3~4의 내용을 MUXA, MUXB에 격납시키기 위한 Routine이고, Instruction Fetch할 때에 실행된다. MUXA, MUXB

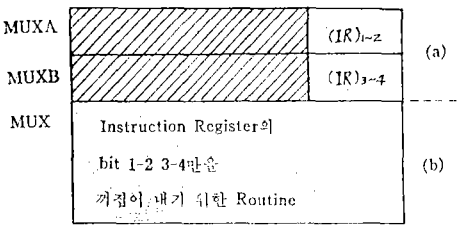

그림 4 A/B 포트선택 走査器의 設定
Fig 4 Design of A/B Ports Select Multiplexer

의 값은 MMI-6701의 General Register 지정에 IR의 bit 1~2, 3~4를 사용할 때 Control ROM Routine에 의하여 각각 A Port, B port에 격납시킨다.

D) 入出力命令의 處理

NOVA Emulator의 入出力機構는 NOVA 1200과 같기 때문에 입출력명령은 Accumulator의 指定, CPU의 Flag Control 지정이 있는것 이외는 User Program의 명령語를 그림 5와 같이 그대로 실행시킨다. PLA에서 I/O命令을 Decode하면 IO_0번지에 Branch한다.

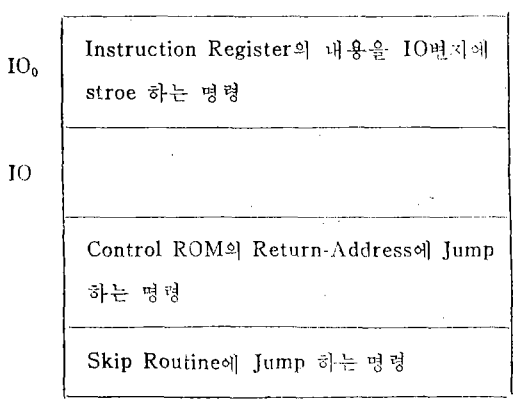

그림 5 I/O명령의 실행방법
Fig. 5 Execution Method of I/O Instruction

Accumulator의 지정이 있는 것은 命令語의 Accumulator 지정을 Zero로 해서 그림 5와 같이 실행시킨다. 이때 假想 Interrupt On Flag만을 Set한다. 또 I.O.F에 의한 브란취命令은 假想 I.O.F에 의하여 실행시킨다.

E) Interrupt處理

NOVA 1200 CPU의 I.O.F는 通常 Clear해 둔다. User Program의 命令의 실행이 終了하면 Interrupt檢出Roututine에 Jump한다.

여기에서는 假想I.O.F.가 Clear되어 있으면 Instruction Fetch에, Set되어 있으면 우선 CPU의 I.O.F.를 Set한다. Interrupt 요구가 없으면 CPU의 I.O.F.를 Clear하고 다음 Instruction Fetch을 한다. Interrupt 요구가 있으면 自動的으로 CPU의 I.O.F를 Clear해서

1番地의 內容에 Jump한다. 1番地에는 Interrupt처리Ruotine의 先頭番地가 들어 있다. Interrupt 처리 Routine에서 Interrupt처리를 한후 Instruction Fetch를 行한다.

Simulator의 List는 Appendix에 附記한다.

4. Emulator에 대한 評價

(1) 實行時間의 比較

Emulator의 Cycle Time을 300ns로 하였을 때 命令實行 時間은 Table 1과 같은 결과를 얻었다.

表 1 에뮤레이터의 實行時間
Table 1 Execution Time of Emulator

Kinds of Instruction		Step No.	Execution Time
Jump		13	5.1us
Jump to Subroutine		14	5.4
Increment or Decrement and Skip if Zero		18	6.6
Load		15	5.7
Store		14	5.4
Arithmetic Instruction	Minimum	16	6.0
	Maximum	29	9.9
I/O Instruction (without Skip)		15	5.7
I/O SKip Instruction		8	3.6

Table 1에서, Arithmetic Instruction의 Minimum Step는 Carry지정, Shift지정이 없을 경우이고, Maximum은 Shift명령으로 Swap를 실행하였을 경우이다. Overflow가 있으면 0.3us (1 Step)追加시킨다.

NOVA 1200의 Cycle Time은 1.2us이므로 Memory 참조명령에서는 약 4倍, 연산명령에서는 약 5~8倍, Skip 명령 이외의 I/O命令에서는 약4倍, I/O Skip명령에서는 3倍가 걸리는 것이 되어 실제는 속도가 2배정도 빠르다.

NOVA의 명령 가운데 연산명령에 대해서는 하나의 命令語에 i) Carry지정, ii)算術論理演算指定, iii) Shift지정, iv) Skip지정, v) No Load or Load 지정의 다섯 종류의 명령이 포함되어 있다. 이 때문에 명령실행에 있어서는 5개의 명령을 順次的으로 Decode하면서 實行하기 때문에 Step數가 大幅으로 증가하게 된다

MMI-67 01에서는 이 다섯개의 명령 가운데 ii), iii), v)는 현재 6 Step으로 실행되지만 1 Step로서 실행할 수도 있다(Shift명령의 Swap는 제외).

이와 같이 하면 實行時間은 短縮되지만 PROM에 格納하는 Step數가 대폭증가 한다.

(2) PLA의 評價

Decode에 PLA를 사용하면 ROM을 사용할 때 보다 매우 간단하게 된다. ROM을 사용해서 Decode하면 入力의 모든 組合에 대해서 출력이 생기지만, 이 PLA를 사용하면 Decode에 필요하지 않은 bit는 Don't Care bit로서 취급하기 때문에 필요한 bit만 Decode할 수 있다.

NOVA의 명령을 Decode하기 위하여 PLA를 사용하면 출력은 110으로 줄 수 있지만, ROM을 사용하면 2^{16}의 출력이 된다. 이와 같이 任意의 Address bit를 사용 가능한 것은 똑같은 결과를 얻는데에 대한 PLA의 利點이 된다.

(3) IC數의 比較

NOVA 1200의 CPU에 사용한 IC는 전부 125개인데 반해서, 設計한 Emulator에서의 IC는 MMI-6701을 포함해 약 40개이다. NOVA 1200과 設計한NOVA EMalator의 IC數를 비교하면 Emulator에서는 約 3분의 1의 IC로서 실현할 수 있었다.

5. 문제점의 考察

(1) 汎用性

이번의 Emulator設計에서는 特定의 計算機(NOVA)만을 Emulation하였지만, 今後의 문제로서는 Emulator의 汎用性을 들 수 있다. 현재의 PROM대신 WCS (Writable Contol Storage)를 사용함에 의하여 Microprogram을 바꾸므로서 다른 명령을 실행시킬 수 있도록 할 필요가 있다.

(2) 入出力의 分散制御

Emulator의 效率化를 위하여 제어를 分散化 시킬필요가 있다. 입출력 장치를 Intelligient化 해서, CPU에 관계없이 입출력동작을 行하는 것 같은 시스템을만드는 것이다. Emulator는 I/O명령이 들어 오면 I/O처리를 한다. 그 사이에 CPU는 Device에 우선권을 빼앗겨 기다려야만 된다. CPU와 주변장치의 動作時間은 CPU ≪Peripheral이기 때문에 주변장치의 영향으로 당연히 Time Loss가 커지게 된다. 따라서 I/O를 위하여 control을 分割하여서 Processor를 사용하므로서 獨自적으로 處理를 할 수 있다면 그만큼 Time Loss가 적게 될 것이다.

6. 結 論

i) 設計한 NOVA Emulator에 필요한 Microprogram量은 약 290 Step이었다.

ii) Emulator의 Simulation을 위한 프로그람量은 1720 Step이었다.

iii) 따라서 전체적으로 Target計算機 보다 염가로 되고, 高速化할 수 있었다. (謝辭) 本研究는 慶應義塾大學의 相磯秀夫博士의 指導下에 의한 것이며 ,同教授에 대해 深深한 謝意를 表합니다. 論文을 作成하는 데에 많은 討論과 助言을 주신 仁荷大의 李柱根博士님께 감사를 드립니다.

<參考文獻>

(1) 宋榮宰 : Microprocessor에 의한 NOVA Emulator의 設計, 電子工學會誌, Vol.13, No.2, pp. 23—28

(2) M.M.Macdougall: Computer System Simulation; An Introduction, Computing Surveys, Vol. 2, No. 3, pp. 191～209, Sept. 1970.

(3) 金澤正憲 : 計算機システムのシミュレションについて, 情報處理, Vol. 13, No. 10, pp. 684～691, Oct., 1972.

(4) Steve Young:A Microprogram Simulator, Proc. of DA Work-shop, 1971, pp. 68～81.

(5) 倉地正 : マイクロプログラムの記述そシミュレション, 情報處理, Vol. 14, No. 6, June 1973, pp. 397～407.

(6) S.S.Husson; Microprogramming: Principles and Pratices. Prentice-Hall, 1970.

〈APPENDIX〉

EXAMPLE OF SIMULATOR LIST

```
        000040          LOC 40
00040 000000 IR:        0
00041 000000 MAR:       0
00042 000000 CF:        0
00043 000000 IOF:       0
00044 00000 MUXA:       0
00045 000000 MUXB:      0
00046 000000 COUNT:     0
00047 000377 SNEXT:     377
00050 003313 SMUX:      MUX
00051 002117 SGRX:      GR
00052 002140 SQR:       Q
00053 002142 SFO:       FO
00054 002143 SOVF:      OVF
00055 002144 SDO:       DO
00056 002145 SDI:       DI
00057 002146 SOP1:      OP1
00060 002147 SOP2:      OP2
00061 002150 SAPRT:     APORT
00062 002151 SBPRT:     BPORT
00063 002141 SCN:       CN
00064 002222 SSLO:      SLO
00065 002227 SSQLI:     SQLI
00066 002230 SSRI:      SRI
00067 002225 SSQRO:     SQRO
00070 002232 SMMI:      MMI
00071 002152 SMMI1:     MMI1
00072 002212 SMMI2:     MMI2
00073 002240 SM2:       M2
00074 002245 SMD:       MD
00075 003070 SFZD:      FZD
00076 003073 SOVFD:     OVFD
00077 003163 SSHL:      SHL
00100 003176 SSHR:      SHR
00101 000000 ROMA:      0
00102 003322 RMM:       MM
00103 060000 MA:        60000
00104 014000 MB:        14000
00105 003211 MPLA:      PLA
00106 003222 MPLA1:     PLA1
00107 003233 MpLA2:     PLA2
00110 003244 MpLA3:     PLA3
00111 003263 MPLA4:     PLA4
00112 003277 MPLA5:     PLAS
00113 003310 MPLA6:     PLA6
00114 000621 RPCIC:     PCINC
00115 000476 RM:        M
00116 000666 RMF:       MF
00117 000546 RQAC2:     QAC3
00120 000531 RQAC2:     QAC2
00121 000514 RQPC:      QPC
00122 000501 RQOO:      QOO
00123 000562 RIND:      IND
00124 000565 RINDR:     INDRT
00125 001116 RAL:       AL
00126 000634 RIO:       MOP
00127 000634 RMOP:      MOP
00130 000635 RMOP:      OPF
00131 001000 RMDSZ:     DSZ
00132 000732 RMISZ:     OISZ
00133 000701 RMJSR:     OJSR
00134 000667 RMJMP:     OJMP
00135 001064 RMSTA:     OSTA
00136 001047 RMLDA:     OLDA
00137 000137 RCC:       RCC
00140 001137 RC1:       C1
00141 001121 RCO:       CO
00142 001173 RALF:      ALF
00143 001355 RAND:      AAND
00144 001255 RAINC:     AINC
00145 001235 RAMOV:     AMOV
00146 001216 RANEG:     ANEG
00147 001176 RACOM:     ACOM
00150 001375 ROVFL:     OVFL
00151 001414 RALSH:     ALSH
00152 001467 RSWAP:     SWAP
00153 001450 RRIGH:     RIGHT
00154 001417 RLEFT:     LEFT
00155 001521 RDNTS:     DONTS
```

```
00156 001533 RALSK:  ALSK
00157 001600 RASBN:  ASEN
00160 001571 RASEZ:  ASEZ
00161 001565 RASNR:  ASNR
00162 001561 RASZR:  ASZR
00163 001555 RASNC:  ASNC
00164 001551 RAAZC:  ASZC
00165 001536 RASKP:  ASKP
00166 001606 RALNL:  ALNLD
00167 001611 RNLOD:  NLOD
00170 001626 RLOAD:  LOAD
00171 001642 RSKIP;  SKIP
00172 001504 RSW:    SW
00173 002026 RINTD:  INTD
00174 000077 RCPU:   CPU
00175 001706 RSDZ:   CS
00176 001706 RSDN:   CS
00176 001706 RSDN:   CS
00177 001706 RSBZ:   CS
00200 001706 RSBNN:  CS
00201 001660 RDOCP:  CDOO
00202 001660 RDOCC:  CDOO
00203 001660 RDOS:   CDOO
00204 001660 RDOC:   CDOO
00205 001663 RDICP:  CDII
00206 001663 RDICC:  CDII
00207 001663 RDICS:  CDII
00210 001663 RDIC:   CDII
00211 001660 RDOBP:  CDOO
00212 001660 RDOBC:  CDOO
00213 001660 RDOBS:  CDOO
00214 001660 RDOB:   CDOO
00215 001663 RDIBP:  CDII
00216 001663 RDIBC:  CDII
00217 001663 RDIBS:  CDII
00220 001663 RDIB:   CDII
00221 001660 RDOAP:  CDOO
00222 001660 RDOAC:  CDOO
00223 00160  RDOAS:  CDOO
00224 001660 RDOA:   CDOO
00225 001663 RDIAP:  CDII
00226 001663 RDIAC:  CDII
00227 001663 RDIAS:  CDII
00230 001663 RDIA:   CDII
00231 001656 RNIOP:  CNIO
00232 001656 RNIOC:  CNIO
00233 001656 RNIOS:  CNIO
00234 001656 RNIO:   CNIO
00235 001706 XSDZ:   CS
00236 001706 XsDN:   CS
00237 001706 XSBZ:   CS
00240 001706 XSBN:   CS
00241 001747 XDOCP:  CHALT
00242 001747 XDOCC:  CHALT
00243 001747 XDOCS:  CHALT
00244 001747 XDOC:   CHALT
00245 001710 XDICP:  CSS
00246 001713 XDICC:  CIORS
00247 001710 XDICS:  CSS
00250 001710 XDIC:   CSS
00251 001727 XDOBP:  CMSKO
00252 001727 XDOBC:  CMSKO
00253 001727 XDOBS:  CMSKO
00254 001727 XDOB:   CMSKO
00255 001717 XDIBP:  CINTA
00256 001717 XDIBC:  CINTA
00257 001717 XDIBS:  CINTA
00260 001717 XDIB:   CINTA
00261 001710 XDOAP:  CSS
00262 001710 XDOAC:  CSS
00263 001710 XDOAS:  CSS
00264 001710 XDOA:   CSS
00265 001737 XDIAP:  CREAD
00266 001737 XDIAC:  CREAD
00267 001737 XDIAS:  CREAD
00270 001737 XDIA:   CREAD
00271 001702 XNIOP:  CNIOF
00272 001677 XNIOC:  CNIOC
00273 001674 XNIOS:  CNIOS
00274 001673 XNIO:   CNIO
00275 002026 XINTD:  INTD
00276 002033 RINT:   INT
00277 000434 RINST:  INSTF
00300 000117 RRAC3:  RQAC3
00013 000131 RRDSZ:  RQDSZ
00302 000135 RRSTA:  RQSTA
00303 000137 RRCC:   RCC
00304 000143 RRAND:  RCAND
00305 000152 RRSWA:  RSWAP
00306 000157 RRSBN:  RRSBN
00307 000175 RRSDZ:  RSDZ
00307 000175 RRSDZ:  RSDZ
03010 000235 RXSDZ:  XSDZ
00311 000000 JIO:    0
00312 002017 JIOIO:  IOIO
00313 001753 JMDO:   MDO
00314 001771 JMDI:   MDI
00315 001670 RDSS:   DSS
00316 001706 RCS:    CS
00317 001715 RIORS:  CIORS
00320 003214 PLOF:   PLOOP
      000602         PRDX2
00321 100000 PL:     1000000000000000
```