internt

EMNE Udvidet lager i RC3600/MUS/DOMUS			Side: 1/1
Udarbejdet af: DHA/TRP	Afd.nr.: 4320	Dato: 19.01.1978	

DISTRIBUTION	
TIL	ADR.
JØB	Rialto
SMN	TUG
SM	TUG
HLV	TUG
MLM	TUG
KDH	TUG
NIA	TUG
OKH	TUG
ERLP	TUG
HLJ	TUG
FVN	TUG
JCO	TUG
KNEH	Århus
AMT	Århus
JCJ	Århus
HEJ	Århus

HH	Glo
JEP	Glo

Hermed et forslag til inkludering af udvidet lager i RC3600/ MUS/DOMUS.

Formålet med udsendelse er udelukkende at kunne få feed-back på de skitserede ideer angående:

1) Anvendelighed med nuværende systemer.
2) Mulighed for hardware implementering.
3) Nye ideer eller ændring til skitsen.
4) Forslag til yderlig performance forøgelse.
5) Omfang af applikations ændringer, hvis løsningen implementeres.

Forslaget ønskes helst kommenteret skriftligt inden den 15.2.1978.

Med venlig hilsen
DHA

Introduktion.

I det følgende gennemgås et forslag til anvendelse af extended memory på RC3600/MUS. Udvidelsen af memory er i princippet en inkludering af de allerede i visse systemer eksisterende ekstra 32kw i 3600-systemet, men indeholder en hardware ændring, der gør det muligt at eksekvere instruktioner i lageret fra 32kw til 64kw.

Formålet med disse sider er ikke at fastlægge den kommende implementering i detaljer, men at delagtiggøre anvendere af 3600/MUS i et forslag og give mulighed for tilbagesvar om huller, begrænsninger og ændringsforslag i den skitserede løsning.

Specielt er de få og nødvendige hardware ændringer kritiske, hvis de foreslåede udvidelser/begrænsninger skaber store problemer i eksisterende program-systemer (test-programmer, store assembler-moduler o.s.v.).

Hardware Ændringer.

a) Indirekte adressering i mere end et niveau fjernes.

D.v.s. at alle adresseberegninger skal medtage alle 16 bit, både når der anvendes indexering eller adressen findes i et ord.

Følgende konstruktion vil da ikke fungere efter tidligere specifikationer:

```
        LDA@   3    PIP ;
PIP:    @PAP            ;
PAP:    10              ;
```

hvor AC3 ikke assignes værdien 10, men indholdet af ord 1b0+PAP.

b) Tilsvarende vil returadressen ved JSR og PC gemt i ord 0 ved interrupt være en 16 bits adresse.

c) MUS GETBYTE, PUTBYTE indføres som hardware instruktion, opererende på bytes i byteadresseområdet 0-64kb.

d) For at skelne diverse CPU-typer skal der implementeres en CPU-identifikations ordre til brug for initialisering.

Funktionen skal defineres, således at ældre CPU'ers manglende svar identificerer disse.

e) Da det må forudses at en række program-systemer udnytter flere niveau indirekte, skal det udvidede lager og funktionerne a) og b) kun være aktive, hvis den gældende RC3603 instruktion for enabling af ext. core er udført.

f) Dette punkt giver en række udvidede hardware funktioner, der kan bidrage til øget performance i fremtidige eller eksisterende MUS-systemer.

For de tidskrævende funktioner må implementeringen i hardware ikke begrænse response på DMA requests.

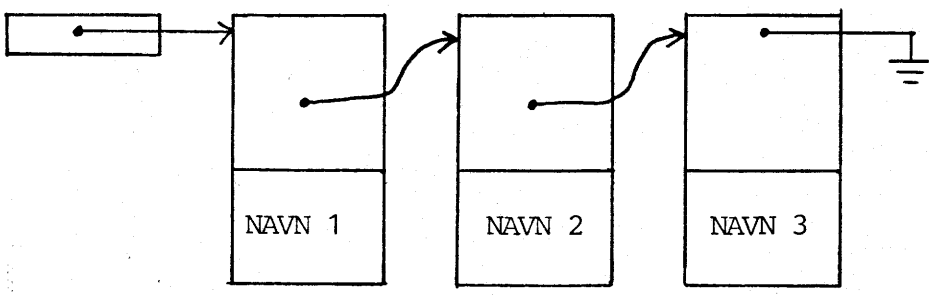
I) Hardware MOVE d.v.s. flytning af bytestrenge. Denne funktion skal desuden kunne interruptes,f.eks. ved at arbejdsvariable er de 4 registre, der gemmes ved interrupt, og at PC først optælles ved afslutning af move operationen, herved eksekveres orderen igen ved interrupt return.

II) Automatisk save af CPU-tilstand ved interrupt, og tilsvarende etablering ved interrupt return.

Den allerede definerede i CPU710 er ikke direkte brugbare i MUS-systemet uden voldsomme software ændringer.

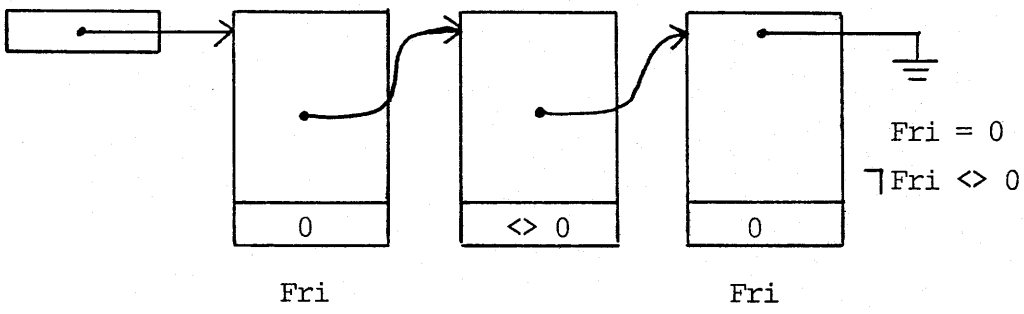
III) Konvertering fra processnavn til processadresse i en enkeltkædet struktur (MUS-SEARCHITEM), som vist på fig. 1.

Fig. 1:



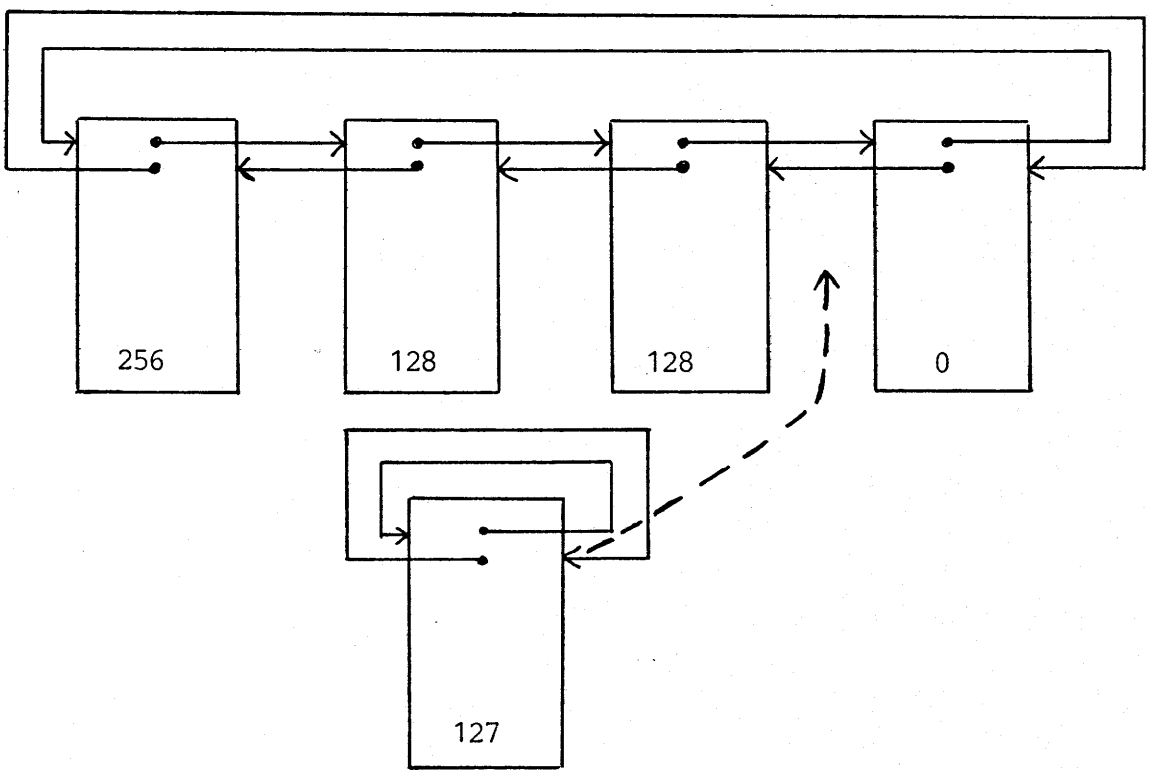
IV) Søgning af frie elementer i enkeltkædet struktur, som vist på fig. 2.

Fig. 2:



V) Indkædning af elementer i dobbelt-kædet struktur direkte eller i prioritetsorden, som vist på fig. 3.

Fig. 3:



VI) Implementering af centrale routiner i MUSIL fortolker specielt adresseberegningsprocedurerne TAKEADDRESS, TAKEVALUE.

VII) Implementering af instruktioner til anvendelse ved generering af div. CRC-checks.

Specielt III, IV og V er tidskrævende i MUS-systemet, hvis der er mange processer,og/eller enkelte processer har megen kommunikation (NCP f.eks.).

I Appendix I er de viste procedurer gengivet med den nuværende software implementering.

Software Ændringer.

I det følgende vil LC angive lower core (0-32kw), og UC angive upper core (32kw-64kw).

Lageradministration.

I DOMUS er der idag en administration af LC med hjælp af core items, hvor alle frie lagerområder er kædet i en enkelt-kædet struktur.

Lagerområder kan reserveres med kommandoen

GET <corename> { <size> } $^{1}_{0}$

hvor udeladelse af <size> giver det største eksisterende core item.

I UC tænkes tilsvarende administreret en kæde af frie lagerområder. Den tilsvarende kommando,der giver et lagerområde i UC kunne være

CETH <corename> { <size> } $^{1}_{0}$.

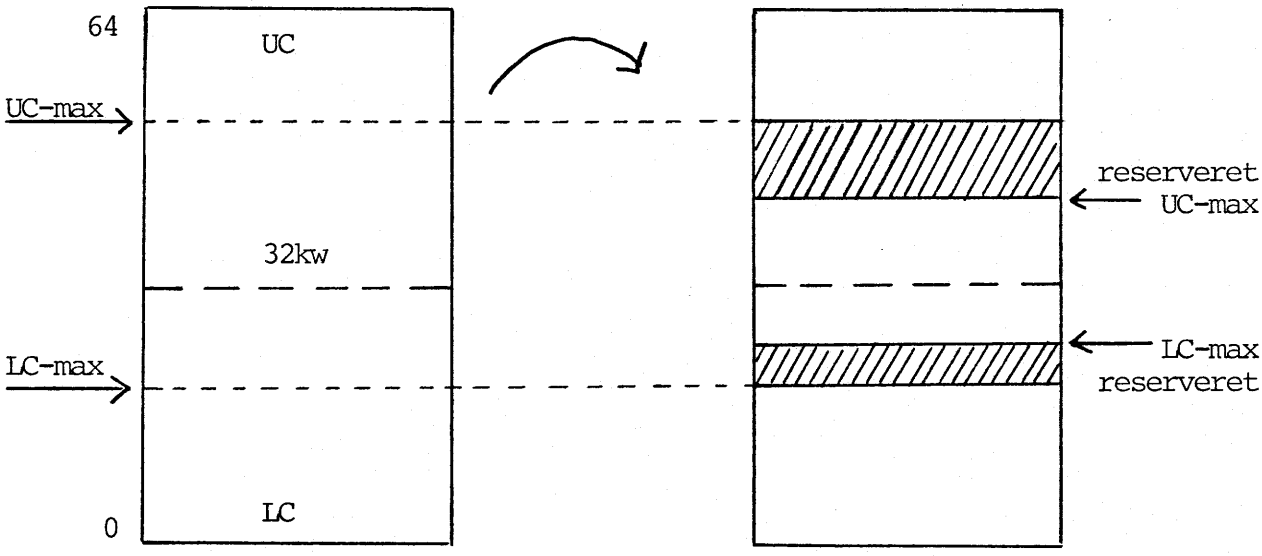
Kan den ønskede størrelse lagerområde ikke findes i UC, søges i stedet i LC, og er området til stede reserveres dette, men der udskrives en "warning" (evt. status, hvis kommandoen er intern).

Funktionen FREE <corename> tænkes uændret, idet begge kæder (UC, LC) scannes for <corename>. Navnesammenfald i UC og LC kæde er derfor ikke tilladt ved GET-kommando.

Den omtalte opsplitning er nødvendig, da byteadresser anvendes overalt i MUS-systemet, og disse dækker kun LC, med mindre specielle procedurer anvendes af applikationen til bytefetch i UC.

I non-DOMUS systemer vil en tilsvarende mekanisme være at foretrække, men af pladshensyn kan det være nødvendigt at anvende en simplere løsning svarende til nuværende stak-mekanisme, på en sådan måde at UC tages fra max-lageradresse med faldende adresser, således at de to stakke vokser mod hinanden (se fig. 4).

Fig. 4:



MUS-Process Ændringer.

I det nuværende system gemmes carry + PC i et 16 bits ord. Med den introducerede hardware ændring skal hardware PC'en være på fuld 16 bit, d.v.s. der skal skaffes plads til en enkelt bit indeholdende carry ved enten at udvide processbeskrivelsen med et enkelt ord, eller at finde "plads" i ubenyttede positioner i processen.

1) Udvidelse af processstørrelsen.

Relativ simpel operation, men kræver uvægerligt at alle moduler recompileres eller editeres og reassembleres, hvis disse er assemblerkodet.

2) Anvendelse af ledig plads.

Det eneste ord i processbeskrivelsen, der ikke anvendes almindeligvis, er size-feltet, og skal nuværende programmer kunne anvendes uændret, må carry stadig hentes i det nuværende PSW process-ord, og size indeholder derfor den mest betydende bit af start-adressen.

Skal det nuværende PSW-ord fortolkes som den fulde 16 bits startadresse, må process opstart defineres. F.eks. vil en minimal ændring være, at alle processer startes af loaderen i den af programmet udpegede start-adresse PSTART. Evt. forskellig opstart kunne styres af en program-kind, da kun loader initialiserer processer, (CAT og XCOM undtaget).

Driver Ændringer.

Standard interrupt clear (CLEAR) sker idag ved to niveaus indirekte adressering, d.v.s. at interrupt clear-proceduren enten må kræves liggende i LC, ellers skal de fleste drivere reassembleres.

Assembler Programmer.

Kode og data, der af programmøren bestemmes til at kunne ligge i UC (ikke byte-adresseret data), udpeges ved et nyt assembler direktiv XREL, svarende til ZREL, NREL.

Der indføres dermed en ny relokeringstype. For at give loaderen nødvendig information til relokeringen indføres desuden en ny bloktype, der indeholder information om det maximalle antal ZREL, NREL og XREL-ord der skal udlægges. Dette gør det muligt at loade NREL og XREL sammenhængende i et område i de tilfælde, hvor UC ikke er tilgængeligt. Samtidig kan anvendelse af lageret optimeres, da loading idag altid foregår i det største eksisterende core item.

Referencer mellem XREL og NREL er underlagt samme restriktioner som nu med ZREL og NREL, d.v.s. kun ordreferencer tillades i assembleren.

Loading vil i princippet da foregå som vist på fig. 5.

Fig. 5a:

Ingen UC

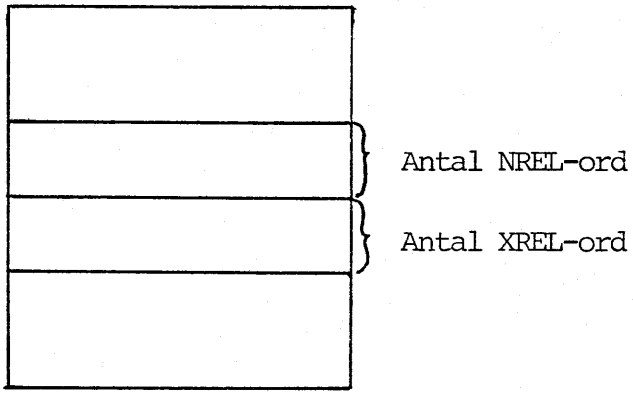
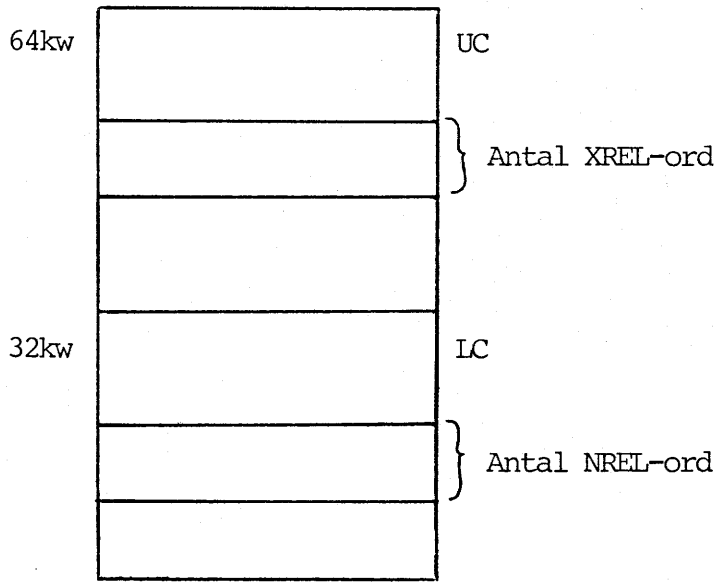


Fig. 5b:

Med UC



Et utility program-load foretages ved reservation af et core item i LC, stort nok til at indeholde det givne antal NREL-ord og et core item i UC (med samme navn) til det givne antal XREL-ord. UC-core item owner sættes til den loadede process, således at oprydning er garanteret ved process KILL.

Skal der foretages load ind i et core item, skal dette ligge i LC, og både XREL og NREL loades i LC, som fig. 5a viser.

I assembleren tænkes desuden udskrift af "warning" ved konstruktionen

@ ADDR

for at fange div. flerniviau indirekte (kan slås fra ved assemblering af pagede programmer).

På programmer og processer vil der være følgende begrænsninger:

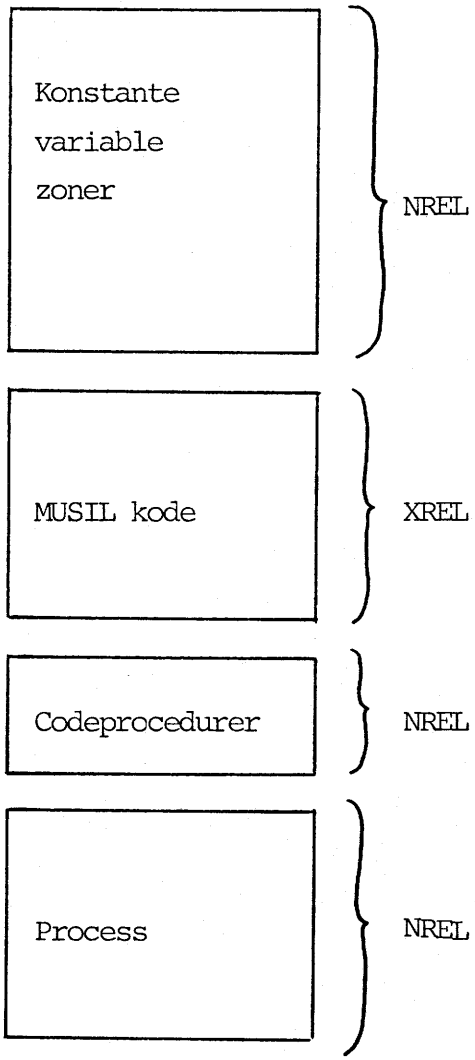
1) Zoner.
2) Processbeskrivelse.
3) Data, der fetches ved GETBYTE/PUTBYTE.
4) Message-buffere
5) Coroutinebeskrivelser og semaphorer

skal være af NREL-type.

MUSIL Programmer.

MUSIL-Compileren tænkes at outputte rel.bin kode i samme XREL, NREL format med følgende fordeling:

Fig. 6:



D.v.s. at kun den rene fortolkbare MUSIL-kode kan loades i UC, hvorved det skønnes at ca. 50% af musil lagerforbruget kan flyttes til UC.

MUSIL-fortolkeren skal af samme grund ændres, således at data kan fetches i UC.

Begrænsninger.

Den væsentligste ulempe ved den skitserede løsning er, at lageret ikke kan antages kontinuert, men er delt i to anvendelighedsmæssigt ret forskellige blokke, d.v.s. at total-udnyttelse af 128 kbytes ikke er muligt.

Desuden vil en afgørelse af om to eller flere større systemer kan loades/eksekveres parallelt ikke afgøres simpelt, idet det reelt er core nok, men systemerne udnytter tilsammen mere af de to lagerhalvdele end de opnåelige 64kw. Denne inflexibilitet kan føre til at hvert enkelt system skal manuelt trimmes til hver aktuel sammensætning af program-systemer for hver lager/system konfiguration. Hermed introduceres de samme administrative ulemper og tilsyneladende manglende lagerudnyttelse, som kendetegner processor-expansion og er derfor kun bekvemt i ret statiske konfigurationer.

UC kan kun i begrænset omfang anvendes til data-buffere, idet kun få drivere kan fetche data over 64 kbytes-grænsen, og i de få tilfælde der gives mulighed for dette, skal ordadresser anvendes, hvilket ikke er understøttet af de normale datahandlings procedurer (zoner).

```
                ; PROCEDURE LINK(HEAD,ELEM);
                ; LINKS A GIVEN ELEMENT TO THE END OF A QUEUE.
                ;         CALL:                 RETURN:
                ; AC0                           DESTROYED
                ; AC1     HEAD                  HEAD
                ; AC2     ELEM                  ELEM
                ; AC3     LINK                  HEAD

00245'054000 A13:  STA     3   0              ; LINK:
00246'135000       MOV     1,3                ;
00247'021401 A130: LDA     0   PREV,3         ;   OLD PREV:= PREV.HEAD;
00250'051401       STA     2   PREV,3         ;   PREV.HEAD:= ELEM;
00251'055000       STA     3   NEXT,2         ;   NEXT.ELEM:= HEAD;
00252'041001       STA     0   PREV,2         ;   PREV.ELEM:= OLD PREV;
00253'053001       STA@    2   PREV,2         ;   NEXT.PREV.ELEM:= ELEM;
00254'002000       JMP@        0              ;   RETURN;

                ; PROCEDURE REMOVE(ELEM);
                ; REMOVES A GIVEN ELEMENT FROM A QUEUE.
                ;         CALL:                 RETURN:
                ; AC0                           DESTROYED
                ; AC1                           UNCHANGED
                ; AC2     ELEM                  ELEM
                ; AC3     LINK                  DESTROYED

00223'054000 A11:  STA     3   0              ; REMOVE:
00224'035000       LDA     3   NEXT,2         ;
00225'057001       STA@    3   PREV,2         ;   NEXT.PREV.ELEM:= NEXT.ELEM;
00226'021001       LDA     0   PREV,2         ;
00227'041401       STA     0   PREV,3         ;   PREV.NEXT.ELEM:= PREV.ELEM;
00230'051000       STA     2   NEXT,2         ;   NEXT.ELEM:= ELEM;
00231'051001       STA     2   PREV,2         ;   PREV.ELEM:= ELEM;
00232'002000       JMP@        0              ;   RETURN;

                ; PROCEDURE LINK PROCESS(PROC);
                ; LINKS A PROCESS TO THE RUNNING QUEUE AS THE LAST PROCESS
                ; AMONG PROCESSES OF SAME PRIORITY.
                ;         CALL:                 RETURN:
                ; AC0                           DESTROYED
                ; AC1                           DESTROYED
                ; AC2     PROC                  PROC
                ; AC3     LINK                  DESTROYED

00233'054000 A12:  STA     3   0              ; LINK PROCESS:
00234'126400       SUB     1,1                ;
00235'045013       STA     1   STATE,2        ;   STATE.PROC:= RUNNING(=0);
00236'034054       LDA     3   RUNNING        ;   HEAD:= RUNNING QUEUE;
00237'021015       LDA     0   PRIORITY,2;
             A120:                            ; NEXT PRIORITY:
00240'035400       LDA     3   NEXT,3          ;   HEAD:= NEXT.HEAD;
00241'025415       LDA     1   PRIORITY,3;     ;   IF PRIORITY.PROC
00242'106432       ING     0,1                ;       <=PRIORITY.HEAD THEN
00243'000775       JMP         A120           ;     GOTO NEXT PRIORITY;
00244'000403       JMP         A130           ;   LINK(HEAD,PROC);
                                              ;   RETURN;
```

```
                ; PROCEDURE SEARCH(CHAIN,NAME ADDR,ITEM);
                ; SEARCHES THE CHAIN FOR AN ITEM WITH A GIVEN NAME AND
                ; DELIVERS IT IF PRESENT, AND A ZERO IF THE NAME IS NOT
                ; FOUND IN THE CHAIN.
                ;         CALL:                 RETURN:
                ; AC0                           DESTROYED
                ; AC1     CHAIN                 DESTROYED
                ; AC2     NAME ADDR             ITEM
                ; AC3     LINK                  CUR

00273'054000 A15:  STA     3   0              ; SEARCH:
00274'135000       MOV     1,3                ;   ITEM:= CHAIN;
             A150:                            ; NEXT ITEM:
00275'035402       LDA     3   CHAIN,3        ;   ITEM:= CHAIN.ITEM;
00276'175005       MOV     3,3 SNR            ;   IF ITEM=0 THEN
00277'000415       JMP         A151           ;     RETURN;
00300'021404       LDA     0   +0+NAME,3      ;
00301'025000       LDA     1   +0,2           ;
00302'106414       INE     0,1                ;   IF 0,NAME,ITEM<>0,NAME ADDR THEN
00303'000772       JMP         A150           ;     GOTO NEXT ITEM;
00304'021405       LDA     0   +1+NAME,3      ;
00305'025001       LDA     1   +1,2           ;
00306'106414       INE     0,1                ;
00307'000766       JMP         A150           ;   IF 1,NAME,ITEM<>1,NAME ADDR THEN
00310'021406       LDA     0   +2+NAME,3      ;     GOTO NEXT ITEM;
00311'025002       LDA     1   +2,2           ;
00312'106414       INE     0,1                ;
00313'000762       JMP         A150           ;   IF 2,NAME,ITEM<>2,NAME ADDR THEN
00314'171000 A151: MOV     3,2                ;     GOTO NEXT ITEM;
00315'034040 A152: LDA     3   CUR            ;
00316'002000       JMP@        0              ;   RETURN;
```

```
                ; PROCEDURE TAKEADDRESS(MODIF,ADDRESS);
                ; GET THE ADDR OF AN INTEGER OR STRING ADDRESSED BY PC AND INCR(P
                ; 2 BITS MODIF: 00 ADDR=WORD(PC). !INTEGER!
                ;     -      -  : 01 ADDR=   -       !STRING!
                ;     -      -  : 10 ADDR=   -       !FILE!
                ;     -      -  : 11 ADDR=   -      .ADDR=ZN(CUR+ADDR(0:7)).ZFIRST
                ;                                        +ADDR(8:15).
                ;         CALL                RETURN
                ; AC0     MODIFBITS           MODIFBITS SHIFT(-2)
                ; AC1                         ADDRESS
                ; AC2     CUR                 CUR
                ; AC3     LINK                DESTROYED
                ;
00423'027033    CL4:    LDA@    1       PC,2    ; ADDRESS:=WORD(PC.CUR);
00424'011033            ISZ             PC,2    ; INCR(PC.CUR);
00425'101223            MOVZR   0,0     SNC     ; BITS:=MODIFBITS EXTRACT 2;
00426'101221            MOVZR   0,0     SKP     ; MODIFBITS:=MODIFBITS SHIFT (-2)
00427'101223            MOVZR   0,0     SNC     ; IF BITS<>11 THEN
00430'001400            JMP             +0,3    ; RETURN;
00431'055025            STA     3       SAVE1,2 ;
00432'034143            LDA     3       .255    ;
00433'137400            AND     1,3             ; FIELD:=ADDRESS(8:15);
00434'166700            SUBS    3,1             ; ADDRESS:=ZN.CUR
00435'133000            ADD     1,2             ;           (ADDRESS(0:7));
00436'031041            LDA     2       ZN,2    ; ADDRESS:=(ADDRESS.ZFIRST)
00437'025017            LDA     1       ZFIRST,2;           +FIELD;
00440'167000            ADD     3,1             ;
00441'030040            LDA     2       CUR     ;
00442'003025            JMP@            SAVE1,2 ; RETURN;

                ; PROCEDURE TAKEVALUE(MODIF,VALUE)
                ; GETS THE VALUE OF AN INTEGER.
                ; 2 BITS MODIF: 00 VALUE=INST(PC); INC(PC) ;
                ;     -      -  : 01 VALUE=P;
                ;     -      -  : 10 VALUE=WORD(INST(PC)); INC(PC);
                ;     -      -  : 11 VALUE=P;
                ;         CALL                RETURN
                ; AC0     MODIF               MODIF SHIFT (-2)
                ; AC1                         VALUE
                ; AC2     CUR                 CUR
                ; AC3     LINK                DESTROYED
                ;
00443'101222    CL5:    MOVZR   0,0     SZC     ;
00444'000413            JMP             CL52    ;
00445'101222            MOVZR   0,0     SZC     ; CASE MODIFBITS(14:15) OF
00446'000404            JMP             CL51    ;
00447'027033            LDA@    1       PC,2    ; 0: VALUE:=INST(PC.CUR):
00450'011033            ISZ             PC,2    ;    INCR(PC.CUR);
00451'001400            JMP             +0,3    ;    RETURN;
00452'031033    CL51:   LDA     2       PC,2    ; 2: VALUE:=WORD(INST(PC.CUR));
00453'027000            LDA@    1       +0,2    ;
00454'030040            LDA     2       CUR     ;
00455'011033            ISZ             PC,2    ;    INCR(PC.CUR);
00456'001400            JMP             +0,3    ;    RETURN;
00457'025032    CL52:   LDA     1       R,2     ; 1 AND 3:
00460'101220            MOVZR   0,0             ;    VALUE:=P.CUR
00461'001400            JMP             +0,3    ;    RETURN;
```