PIOTR STRZAŁKOWSKI
Instytut Energii Atomowej – CYFRONET
Świerk k/Warszawy
STANISŁAW WÓLTAŃSKI
Przemysłowy Instytut Automatyki
i Pomiarów MERA–PIAP
Warszawa

SYSTEMY OPERACYJNE CZASU RZECZYWISTEGO DLA MIKROKOMPUTERÓW 16 – BITOWYCH
– charakterystyka porównawcza

Praca zawiera krótką charakterystykę porównawczą kilkunastu systemów operacyjnych czasu rzeczywistego – od minimalnego jądra dla stacji lokalnej do dużego systemu obsługującego sieć mikrokomputerów. Zasygnalizowano kierunki rozwoju w tej dziedzinie. Zamieszczono obszerny wykaz literatury związanej z tematem.

1. Podstawowe pojęcia

System operacyjny jest to oprogramowanie podstawowe stanowiące sprzężenie między oprogramowaniem aplikacyjnym a sprzętem komputerowym. Jego główną funkcją jest zapewnienie efektywnego zarządzania dostępem wielu użytkowników do wspólnie wykorzystywanych zasobów (użytkownikami mogą tu być zarówno ludzie pracujący przy terminalach jak i różne programy starające się jednocześnie o dostęp do centralnego procesora).

System operacyjny zna cały dostępny sprzęt, wie jak z niego korzystać, jak zapewnić przesłanie informacji między zasobami oraz jak wykorzystywać programy użytkownika. W otoczeniu wielozadaniowym musi zapewnić koordynację wykonywania poszczególnych zadań (czy procesów). System operacyjny zarządza więc czasem procesora, różnego rodzaju urządzeniami, wykonywanymi procesami, pamięcią, zapewnia obsługę we/wy systemu plików, zarządza zasobami oprogramowania, zapewnia komunikację z użytkownikiem (m.in. interpretuje polecenia wprowadzane z terminala) i korzystanie z oprogramowania użytkowego.

Mikrokomputerowe systemy operacyjne można podzielić na dwa rodzaje:

- systemy ukierunkowane na obsługę procesu tworzenia i wykonywania programu, w których główną rolę odgrywają systemy zarządzania, edytory, asemblery, kompilatory i programy łączące;

- systemy operacyjne czasu rzeczywistego, ukierunkowane na obsługę zewnętrznych procesów i działające według ściśle określonych kryteriów czasowych.

W tej drugiej kategorii zewnętrzne zdarzenia powodują uaktywnienie pewnych zadań. Działania wykonywane są w otoczeniu, w którym fizyczne zdarzenia mają charakter losowy a fizyczne procesy występują równolegle. Stąd dwie charakterystyczne cechy systemów uwarunkowanych czasowo: wielozadaniowość i umiejętność zapewnienia obsługi przerwań.

Poza powyższym podziałem można wyróżnić trzeci rodzaj systemów, tzw. systemy interakcyjne, także wielozadaniowe i współbieżne, w których nie musi jednak występować obsługa przerwań (np. systemy rezerwacji miejsc, systemy bankowe). Zasadnicza różnica pomiędzy systemami R—T a interakcyjnymi polega na roli tzw. czasu reakcji (ang. response time). W systemach R—T czas reakcji, dla najgorszego przypadku, nie może przekroczyć żądanego i gwarantowanego ograniczenia. Nie oczekuje się odpowiedzi maksymalnie szybkiej. W systemach interakcyjnych przeciwnie: czas odpowiedzi powinien być maksymalnie krótki; dopuszczalne są natomiast przypadki dłuższej zwłoki, jeżeli nie zdarzają się zbyt często. W systemach zawierających zarówno uwarunkowania czasowe jak i interakcyjność należy przede wszystkim zagwarantować spełnienie tych warunków, a następnie optymalizować czas odpowiedzi interakcyjnej. Najprostszym rozwiązaniem, ale nie zawsze optymalnym, jest nadanie zadaniom czasu rzeczywistego wyższych priorytetów niż zadaniom interakcyjnym.

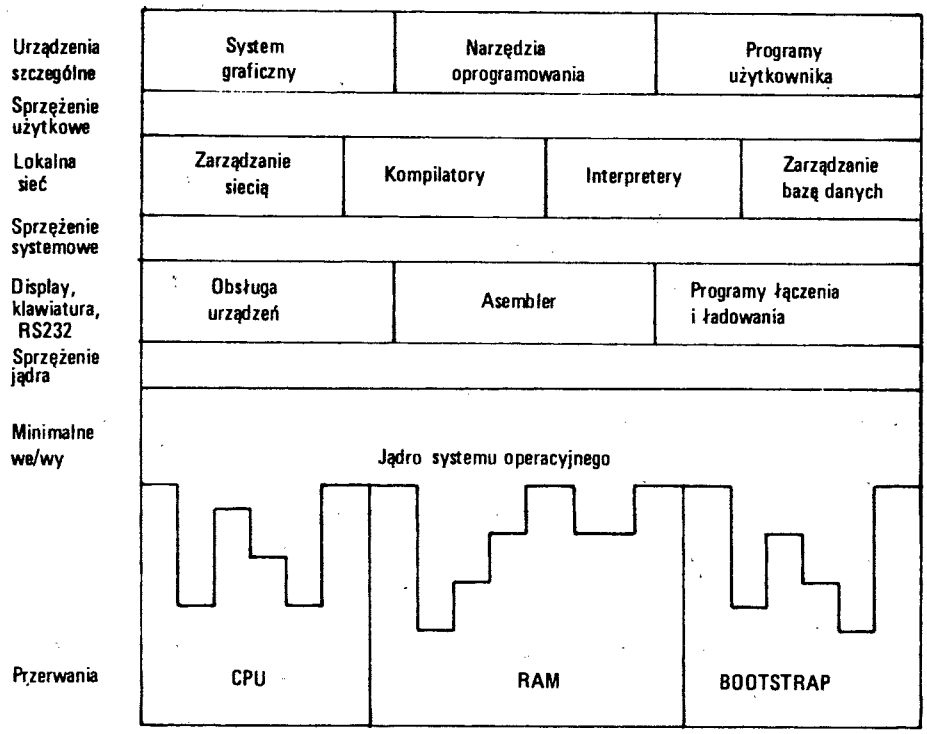
System operacyjny i oprogramowanie użytkowe stanowią osobne warstwy oprogramowania; sam system operacyjny może być również zbudowany z warstw.

Rysunek 1 przedstawia strukturę i sprzężenia mikrokomputerowego systemu operacyjnego.

Komunikacja użytkownika z systemem operacyjnym jest możliwa w oprogramowaniu użytkowym poprzez wywołania funkcji systemowych (będziemy tu czasem używali zamiennie terminu „ekstrakody"). Zależnie od stopnia złożoności systemu funkcji takich może być od kilkunastu do kilkuset.

Jednym z najważniejszych elementów systemu operacyjnego czasu rzeczywistego jest mechanizm obsługi wielozadaniowości, realizowany przez koordynator zadań. Pozostała część obudowuje to „jądro" i jest odbiciem wymagań stawianych przez otoczenie aplikacyjne. W szczególności, w celu współdziałania z różnymi zasobami fizycznymi systemu tworzy się kolejki zawierające zadania (procesy lub programy w trakcie wykonywania) oczekujące na podjęcie wykonywania przez procesor, na dostęp do sprzętu we/wy czy np. na zajście określonego zdarzenia. Są to osobne kolejki, z których koordynator, według określonego algorytmu szeregowania zadań, wybiera zadanie i zezwala na jego wykonanie przez jednostkę centralnego procesora. Wybór może być oparty na prostej technice FIFO (obsługa według kolejności wejścia do kolejki: pierwszy, który wszedł zostaje pierwszy obsłużony), na pobraniu kolejnego zadania z kolejki „koło-

wej" i przydzieleniu mu kwantu czasu centralnego procesora, po którym zadanie zostaje wprowadzone ponownie na koniec kolejki, na systemie priorytetów (zadanie o wyższym priorytecie wypiera zadania o niższym priorytecie), czy też na kombinacji tych technik.

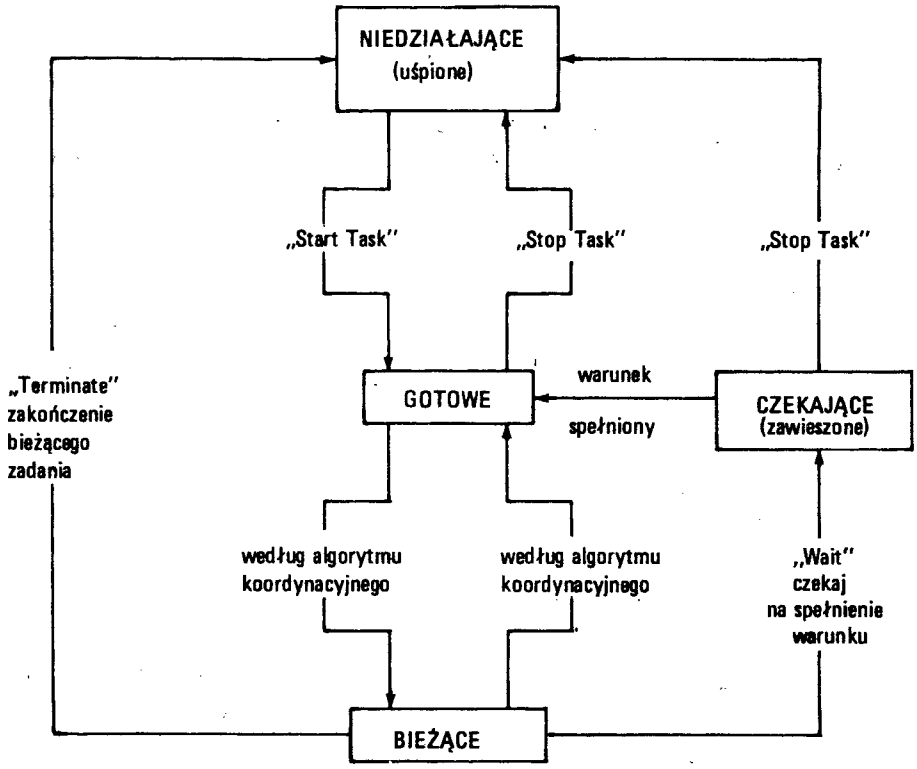


Rys. 1. Struktura i sprzężenia uniwersalnego, mikrokomputerowego systemu operacyjnego. Przykładem takiego systemu może być MS–DOS IBM PC, wykorzystywany w pracach nad SIRTOS–em.

W algorytmach koordynujących często zakłada się, że zadania mogą być w pewnej liczbie stanów logicznych, zależnie od ich gotowości do wykonywania. Przykładowo, mogą istnieć 4 rodzaje stanów: bieżącego wykonywania (aktualnie wykonywane zadanie), uśpienia przez wywołanie systemowe, zawieszenia do momentu uzyskania dostępu do zasobu i gotowości do wykonania. W odpowiedni stan zadanie może być wprowadzone przez zdarzenia zewnętrzne, przez inne zadanie i przez systemowe programy

usługowe lub przez wywołanie z zewnątrz zadania odpowiedniej funkcji systemowej. Przykładowe stany zadania przedstawia rysunek 2.

W systemach operacyjnych czasu rzeczywistego istnieje także możliwość zawieszania zadań na określony kwant czasu(zadawany poprzez licznik cykli zegara), które tworzą wówczas osobną kolejkę.



Rys. 2. Stany zadań w systemie „REALTIME CRAFT" (por. p. „Przykładowe systemy"). W cudzysłowach wyróżniono wywołania funkcji systemowych.

Systemy operacyjne czasu rzeczywistego wymagają również stworzenia zabezpieczeń przed sytuacjami konfliktowymi podczas przydziału zasobów, np. wtedy, gdy dwa zadania żądają tego samego zasobu. Do tego celu wykorzystywane są zwykle semafory — mechanizm polegający na blokowaniu dostępu do zasobu w momencie użycia go przez któreś z zadań i umieszczeniu kolejnych zadań pragnących uzyskać do niego dostęp w kolejce oraz na odblokowaniu tego dostępu w momencie zwolnienia zasobu przez

zadanie. Wykorzystuje się także tzw. monitory będące realizacją obszarów krytycznych. Semafory można wykorzystywać również do synchronizacji zadań wykonujących szereg czynności sterujących procesem (w celu realizacji np. określonej sekwencji sterowania poszczególnymi układami funkcjonalnymi obiektu).
W przypadku, gdy między zadaniami komunikacja jest bardziej złożona niż proste przekazywanie sygnałów zajętości, np. gdy przekazywane są między nimi dane, wykorzystuje się tzw. skrzynki pocztowe, które są rodzajem semaforów z pamięcią. Zadanie potrzebujące odpowiedniej informacji czeka do momentu, w którym inne zadanie wypełni nią właściwą skrzynkę pocztową.
Osobnymi kwestiami są: uwzględnienie otoczenia interakcyjnego, w którym trzeba liczyć czas reakcji systemu na wciśnięcie klawisza, sposób obsługi przerwań czy wyszukiwanie błędów w programach użytkowych i zabezpieczenie przed nimi systemu. W bardziej rozbudowanych systemach uwzględnia się wszystkie problemy związane z obsługą we/wy, np. zarządzanie plikami.
Obszerny opis przedstawionych tutaj pobieżnie cech systemów operacyjnych, pojęć i technik z nimi związanych, można znaleźć w podanej na końcu literaturze.
Powyższy opis wybranych podstawowych cech mikrokomputerowych systemów operacyjnych czasu rzeczywistego zostanie uzupełniony obecnie kilkoma dodatkowymi uwagami.

2. Geneza i kierunki rozwoju

Systemy uwarunkowane czasowo istniały na długo przed pojawieniem się techniki mikroprocesorowej. Większość pojęć i technik opisanych w punkcie „Podstawowe pojęcia" została wypracowana wcześniej. Mikroprocesory spowodowały jednak (m.in. dzięki dużemu potanieniu sprzętu) znaczne przyspieszenie rozwoju zastosowań, w tym także w zastosowaniach związanych z czasem rzeczywistym (przetwarzanie transakcji, sterowanie procesami przemysłowymi, zastosowania wojskowe i inne), szerokie upowszechnienie techniki komputerowej, pojawienie się wielu „inteligentnych" urządzeń we wszystkich dziedzinach życia. Jednocześnie do techniki i oprogramowania mikrokomputerowego wprowadzono szereg wcześniej wypracowanych koncepcji informatycznych (związanych np. z otoczeniem programistycznym itp.).
Pierwszym systemem operacyjnym dla mikroprocesorów był CP/M dla 8–bitowego mikroprocesora Intel 8080, który pojawił się około roku 1978. Około roku 1981 zapowiedziano pierwsze systemy operacyjne dla procesorów 16–bitowych. Wśród tych, które pojawiły się np. CP/M 86, MS–DOS, p–code, system UCSD, MSP/Z8000 czy iRMX 86, część była nastawiona na zastosowania związane z czasem rzeczywistym. Wiele z tych systemów zawierało nowe pomysły, niektóre z nich zostaną omówione w następnym rozdziale. Systemy takie jak VRTX, iRMX 86, MSP/Z8000 czy ZRTS

były pisane całkiem od początku. Jednocześnie jednak zaznaczyła się inna tendencja: oparcie się na wcześniejszych, zdobywających coraz szerszą popularność systemach. Systemem takim był Unix, utworzony do innych zastosowań niż uwarunkowane czasowo, ale prędko przeniesiony na mikrokomputery i rozbudowany m.in. do wykorzystania w czasie rzeczywistym. Przyczyną tego stanu rzeczy były dwa czynniki potraktowanie Unixa jako otoczenia programistycznego, w którym zastosowano wygodne i oryginalne rozwiązania oraz fakt, że język C, w którym i wokół którego system zbudowano ocenia się jako jeden z najbardziej odpowiednich języków do zastosowań uwarunkowanych czasowo. Odpowiedniość ta nie polega zresztą na istnieniu w nim jawnie mechanizmów czasu rzeczywistego (np. konstrukcji umożliwiającej współbieżność wykonania) ale na łatwości z jaką można przechodzić na niższy poziom, na którym mechanizmy te można tworzyć. Obecnie istnieje cała rodzina systemów operacyjnych czasu rzeczywistego, opartych na Unixie, a sam system wydaje się być najbliżej standardu w tym zakresie.

Wśród zjawisk związanych z mikroprocesorowymi układami czasu rzeczywistego ostatnich lat, należy odnotować pojawienie się tzw. „oprogramowania krzemowego" (ang. silicon software, siliconware), umieszczanego w pamięci ROM i mającego zastosowanie głównie w systemach wbudowanych (ang. embedded), o minimalnej konfiguracji sprzętowej z mikroprocesorami sterującymi fragmentami większego systemu, tworzone na zewnątrz niego. Samo umieszczenie w pamięci ROM jednak nie wystarczy, aby oprogramowanie mogło zostać nazwane „krzemowym". Jest to określenie zarezerwowane dla programów w pamięci ROM, które mogą być użyte jako składowe oprogramowania w sposób podobny do tego, w jaki obwody scalone TTL serii 74 są używane w inżynierii sprzętowej. Użytkownik nie zna wewnętrznego działania urządzenia, zna tylko jego funkcje i protokóły we/wy, co jest równoważne specyfikacji elektrycznej i tablicom logicznym obwodu scalonego.

„Krzemowe" systemy operacyjne rozszerzają zbiór rozkazów mikroprocesora np. o elementarne operacje (ang. primitives), wykonujące działania na kolejkach czy tablicach a nie na rejestrach sprzętowych. W niektórych systemach dodawany jest specjalny zbiór rozkazów obsługi przerwań. Możliwe jest także logiczne rozszerzanie systemu operacyjnego przez użytkownika.

Warto również wspomnieć o ułatwieniach związanych z projektowaniem i programowaniem systemów czasu rzeczywistego (część z nich powstała jeszcze w latach 70.): wprowadzeniu odpowiednich konstrukcji do języków programowania wyższego poziomu takich jak C, Forth, ADA, Modula, Pearl, Concurrent Pascal, Gypsy, modyfikacje Simuli czy nawet APL, umożliwieniu opisu oraz jego analizy na poziomie projektowania w systemach zbliżonych do oprogramowania ISDOS Project i innych. Częściowo wiąże się to z całościowym potraktowaniem procesu tworzenia i eksploatacji programu w otoczeniach programistycznych, takich jak Unix czy APSE.

W ostatnich latach powstało wiele nowych języków programowania ukierunkowanych na realizację współbieżności, zarówno w systemach scentralizowanych jak i rozproszonych. Najbardziej znane z nich to:

- Euklides – wykorzystany do opracowania systemu operacyjnego Tunis (funkcjonalnego odpowiednika systmu Unix),
- Edison – przewidziany do programowania systemów rozproszonych (wykorzystujący niektóre elementy języków Pascal, Concurrent Pascal oraz Modula 2),
- Occam – przystosowany do obsługi sieci złożonej z dużej liczby jednakowych i prostych mikroprocesorów (określany często jako asembler dla transputerów),
- Joyce – modelowy język programowania współbieżnego z notacją Pascala.

Interesujące są także próby wprowadzenia, w zastosowaniach czasu rzeczywistego, systemów z bazami wiedzy (ang. expert systems), umożliwiających szybką analizę i reagowanie na złożone zdarzenia zewnętrzne z silnymi ograniczeniami czasowymi. Próby takie prowadzone są m.in. w RFN.

3. Przykładowe systemy

Obecnie wymienimy niektóre z 16–bitowych systemów operacyjnych czasu rzeczywistego, wraz z krótką informacją o ciekawszych rozwiązaniach zastosowanych w tych systemach.

3.1. VRTX (Versatile Real–Time Executive) firmy Hunter & Ready Inc z Palo Alto, Kalifornia

Jest zbiorem niezależnych od procesora bloków, które programiści mogą użyć do budowania pakietów aplikacyjnych dla systemów wbudowanych. Utworzenie oprogramowania procesora docelowego dokonuje się w osobnym systemie.

Zawiera tzw. tablice konfiguracji w systemowej pamięci RAM, umożliwiającej wprowadzenie rozszerzeń systemu operacyjnego przez użytkownika (np. dołączenie oprogramowania wyszukiwania błędów czasu wykonania). W tablicy tej umieszcza się specyfikację procedury użytkownika, któa jest uruchamiana podczas inicjacji systemu. Można też umieścić w niej wskaźniki specyfikujące procedury dostępne w czasie tworzenia i usuwania zadania lub wtedy, gdy uzyska się wymagany kontekst. Można również rozszerzyć zarządzanie plikami.

Dzięki wprowadzeniu w „krzem" rozszerzeń, dysponuje specjalnym zbiorem rozkazów obsługi przerwań.

Cztery możliwe stany zadań: wykonywane, uśpione, zawieszone i gotowe do wykonania.

Oferowany jest w zestawie: VRTX + IOX (we/wy) + FMX (zarządzanie plikami).

3.2. iRMX 86 firmy Intel Corporation z Santa Clara, Kalifornia

Zawiera, jako krzemowe jądro, procesor 80130. Inne krzemowe oprogramowanie Intela to: 82730 — koprocesor tekstowy, 82583 — koprocesor lokalnej sieci, 82720 — procesor graficzny. W odróżnieniu od VRTX, jądra krzemowe są podzbiorami pełnego systemu operacyjnego i nie wymagają osobnego systemu do tworzenia oprogramowania mikroprocesora docelowego (poza iRMX 86).

Ma strukturę opartą na kontekście. Każde zadanie znajduje się w kontekście otoczenia pracy (ang. job environment), ograniczającym zakres znajdujących się w nim zadań. Obiektami w takim otoczeniu są zadania, skrzynki pocztowe i semafory. Zadanie może korzystać jedynie z pamięci dostępnej w danym otoczeniu. Gdy potrzeba więcej pamięci, wykonuje się odpowiednie wywołania systemowe zwiększające przydział pamięci dla danego otoczenia.

Procesor 80130 uogólnia koncepcje tablicy konfiguracji z VRTX, wprowadzając tablicę indeksów zawierającą wskaźniki do procedur systemowych. Procedury systemowe można wymieniać. W systemie wbudowanym nowa procedura może zostać umieszczona w pamięci ROM razem z oprogramowaniem aplikacyjnym.

Posiada, podobnie jak VRTX, w wirtualnym mikroprocesorze specjalny zbiór rozkazów obsługi przerwań.

Zapewnia obsługę sytuacji wyjątkowych (posiada tzw. exception handler) oraz wyszukiwanie błędów (debugging), w tym badanie poszczególnych zadań i nadzorowanie działania systemu oraz sprawdzanie obiektów systemowych.

Zawiera tzw. „fizyczny plik", w którym sektory dyskowe występują jeden po drugim. Kolejny blok danych jest zapisywany w następnym fizycznym sektorze dysku, co zmniejsza opóźnienia spowodowane ruchem głowicy. Daje również możliwość stosowania tzw. „przeplotu" z wybieranym podczas formatowania nośnika współczynnikiem, określającym co ile rekordów fizycznych leży kolejny rekord logiczny. Różne rodzaje przetwarzania zbioru (kopiowanie, ładowanie do pamięci) wymagają różnego współczynnika „przeplotu" w celu minimalizacji czasu wykonania.

Rozszerzenie iRMX (tzw. MMX — Multibus Message Exchange) dla systemów wieloprocesorowych wprowadza zamiast zwykłych skrzynek pocztowych z iRMX tzw. porty logiczne (ang. software ports), umożliwiające komunikację między zadaniami niezależnie od tego, czy są one w tym samym, czy fizycznie innym procesorze.

Zawiera osobne poziomy oprogramowania używane do przetwarzania przerwań (tzw. interrupt handler na niskim poziomie i tzw. interrupt task na poziomie wyższym).

W kolejce zegarowej nie zmienia cyklu zegara wszystkich liczników, ale sprawdza tylko poprzedni element kolejki, zmieniając jeden licznik.

Komunikacja z urządzeniami peryferyjnymi jest niezależna od tych urządzeń i wykonywana za pomocą pewnej liczby funkcji systemowych.

3.3. MSP/Z8000 firmy Hemenway Corporation z Bostonu, Massachusetts

Podobnie jak w iRMX, krzemowe jądro stanowi część pełnego systemu operacyjnego.

Realizuje obszary krytyczne poprzez „monitory" za pomocą czterech funkcji: Entermon, Exitmon, Wait i Signal. Pierwsze dwa służą jako wejście i wyjście z monitora (wstrzymanie przerwań i zachowanie rejestrów na wejściu i działania odwrotne na wyjściu). Wait i Signal działają jako operacje na semaforach.

Zawiera tzw. „przyległy plik" (ang. contiguous file), odpowiednik „fizycznego pliku" w iRMX.

Używa jednej podstawowej procedury we/wy, obsługującej wszystkie operacje we/wy dzięki dostępowi do specjalnego bloku informacji w pamięci (w odróżnieniu np. od iRMX).

3.4. ZRTS firmy Zilog Corporation z Cupertino, Kalifornia

Podobnie jak w VRTX, konieczne jest użycie osobnego systemu do tworzenia oprogramowania docelowego mikroprocesora.

Posiada język konfiguracji, w którym, definiując szczegóły dotyczące sprzętu, projektant systemu może utworzyć konkretny system będący konfiguracją ZRTS.

3.5. REALTIME CRAFT francuskiej firmy TECSI

Część wykonawcza XEC oparta jest na standardzie SCEPTRE i określana jako składnik krzemowy. Cały system oferowany jest w zestawie:

XEC + IOS (we/wy) + FMS (system zarządzania plikami)

Posiada 27 funkcji systemowych. Wykorzystuje do ochrony zasobów koncepcje obszarów krytycznych i semafory, do komunikacji wykorzystuje się też skrzynki pocztowe.

Zapewnia obsługę sytuacji wyjątkowych.

Jest niezależny od systemu, w którym tworzy się oprogramowanie na docelowy mikrokomputer.

Formaty dyskowe są zgodne z formatami Unixa.

3.6. SI firmy Multi Solutions Incorporation z Lawrenceville, New Jersey

Uniwersalny z możliwością działania w czasie rzeczywistym. Zawiera różne algorytmy szeregowania zadań w systemie z przerwaniami: najprostszy algorytm przydziela wszystkim zadaniom równe porcje czasu, a wszystkie aktywne zadania wykonywane są kolejno. Bardziej skomplikowany algorytm uwzględnia priorytety, a algorytm najbardziej złożony dzieli zadania na interakcyjne i wsadowe (te, które nie przesyłają żadnych informacji na terminal). Algorytm dokonuje analizy zachowania się zadania, tak aby zadania interakcyjne miały zapewniony pożądany czas odpowiedzi.

Obsługa zadań asynchronicznych jest możliwa za pośrednictwem definiowanych warunków, określanych m.in. przez akcję podejmowaną przez system w momencie wystąpienia zdarzenia. Warunki stosuje się głównie do sygnalizacji błędów i zdarzeń wyjątkowych.

Synchronizacja międzyzadaniowa jest realizowana poprzez semafory.

3.7. Concepts firmy Bell Laboratories

Concepts nie jest systemem operacyjnym, ale systemem sterowania procesami. Został opracowany w Bell Laboratories w Murray Hill, N.J. Jest on przykładem architektury uwzględniającej system operacyjny Unix. Wykorzystuje komputer (PDP 11/23), na którym działa zwykły Unix oraz pomocnicze mikroprocesory (LSI–11) obsługujące działania czasu rzeczywistego niskiego poziomu, z oprogramowaniem będącym rozszerzeniem Unixa.

3.8. Executive firmy JMI Software Consultants z Roslyn, Pensylwania

Jest rozszerzeniem biblioteki wykonawczej języka C o cechy konieczne w zastosowaniach czasu rzeczywistego. Wymiana danych między zadaniami następuje poprzez mechanizm kolejek. Koordynacja zadań oparta jest na priorytetach.

Przeznaczony jest przede wszystkim dla systemów wbudowanych, tj. bez dysków i jest całkowicie zawarty w pamięci systemowej. Nie zajmuje się zarządzaniem plikami.

3.9. Venix firmy VenurCom z Cambridge, Massachusetts

Oparty na Unixie. Koordynacja zadań polega na manipulacji w Unixie poziomami priorytetów, przy czym koordynator pozwala na wykonywanie się specjalnej klasy zadań na poziomie priorytetowym czasu rzeczywistego tak długo jak jest to konieczne.

Do konwencjonalnej, synchronicznej procedury we/wy Unixa, w której zadanie żądające we/wy zawiesza się do momentu wykonania się operacji, dodawana jest procedura asynchroniczna. Zadania asynchroniczne są wprowadzane na początek kolejki zadań we/wy. Zadanie czasu rzeczywistego żąda np. zapisu i wykonuje się dalej mając pewność, że jego żądanie będzie później spełnione.

3.10. Regulus firmy Alcyon Corporation z San Diego, Kalifornia

Oparty na Unixie. Przyśpiesza reakcje na zdarzenia czasu rzeczywistego za pomocą 32 definiowanych przez użytkownika sygnałów priorytetowych.

3.11. Unix System III (zmodyfikowana wersja) firmy Masscomp z Littleton, Massachusetts

Dodany jest tzw. mechanizm AST (Asynchronous Signal Trap), umożliwiający zadaniom wykonywanie operacji uzupełniających po wykonaniu właściwej pracy, np. przekazanie informacji do innego zadania o wypełnieniu bufora.

Podobnie jak w MSP/Z8000 koncepcja „przyległego pliku" została zrealizowana w systemie zarządzania plikami.

Standardowy mechanizm komunikacji między zadaniami (tzw. „pipes") w Unixie został rozszerzony o możliwość przesyłania buforów.

3.12. Unos firmy Charles River Data Systems z Natick, Massachusetts

Oparty na Unixie. Niezależnie skonstruowane zadania mogą używać wspólny zbiór obszarów pamięci w celu przesyłania między sobą danych lub w celu wykonywania ciągu obliczeń.

W celu uniknięcia problemów związanych z zadaniami asynchronicznymi został wprowadzony mechanizm tzw. „liczenia zdarzeń" (dokładniej tzw. „event counts") wykonywanego dla poszczególnych zadań. „Liczenie zdarzeń" wykorzystuje się też jako elementarne operacje synchronizacji (wprowadzonej przez semafory) i wzajemnej wyłączności dostępu do zasobów (wprowadzonej przez monitory).

3.13. ARTEX – Advanced Real Time Executive. The Norwegian Inst. of Technology, Trondheim, Norwegia

Przeznaczony dla systemów rozproszonych. Idea zadań wzorowana na modelu zastosowanym w Industrial Real Time Fortran (IRTF). Zastosowanie – od małych stanowisk laboratoryjnych do dużych obiektów przemysłowych.

Pracuje w systemach hierarchicznych zarówno z punktu widzenia sprzętu jak i oprogramowania. Stanowi program użytkowy uniwersalnego systemu operacyjnego GPOS (np. MS–DOS, C–DOS, CCP/M lub iRMX), zainstalowanego w stacji nadrzędnej. Wersja pełna rezyduje w stacjach nadrzędnych, wersje okrojone w podrzędnych (które stanowią zazwyczaj 16–bitowy procesor, komunikujący się ze stacją nadrzędną przez sterowaną przez nią magistralę).

Zadania obsługiwane przez ARTEX są niewidoczne dla systemu GPOS. Istnieje tzw. zewnętrzny i wewnętrzny model stanów zadań. Stany zadań w modelu zewnętrznym: nieistniejące, uśpione, czekające, zawieszone i bieżącego wykonywania, czyli biegnące. Stany zadań w modelu wewnętrznym: wprowadzone, gotowe, zablokowane i wykonywane (stany te realizują stan bieżącego wykonywania z modelu zewnętrznego).

Komunikacja między zadaniami wykorzystuje tzw. globalny numer zadania. Przesyłana informacja jest niewidoczna dla zadań pośredniczących. Producent może oczekiwać na potwierdzenie odebrania informacji przez konsumenta lub nie. Każde zadanie ma własny bufor na informacje.

Językiem źródłowym systemu jest C, uzupełniony o nieliczne wstawki asemblerowe. Programy użytkowe mogą być pisane w dowolnym języku wyższego poziomu. Komunikacja zadań użytkowych z systemem przez mechanizm przerwań logicznych, wymiana danych przez stos.

3.14. SIRTOS – a Small Industrial Real–Time Operating System, MERA–PIAP, Warszawa

Podobnie jak Executive firmy JMI stanowi jądro zawierające mechanizmy synchronizacji i komunikacji dla zadań użytkowych. Można je traktować jako rozszerzenie języka C o mechanizmy czasu rzeczywistego.

Zawiera 13 funkcji systemowych (tzw. „ekstrakodów"), których wykonanie może (po odwołaniu się do koordynatora zadań) zmienić zadanie aktywne, zgodnie z priorytetową kolejką zadań gotowych (stany zadań: uśpione, zawieszone, gotowe, aktywne).

Komunikacja przez skrzynki pocztowe zrealizowane w formie buforów cyklicznych, obsługiwanych przez dwie pary ekstrakodów. Synchronizacja zadań przez semafory i monitory.

Zawiera uniwersalny monitor przerwań z możliwością wielopoziomowej obsługi i opcjonalnym powoływaniem koordynatora zadań. Posiada również specjalny mechanizm obsługi limitów czasowych (ang. timeout). Pomocnicze funkcje systemowe (7) pozwalają m.in. blokować układ przerwań lub maskować wybrane źródło przerwania (i odwrotnie).

3.15. QNX (QUNIX) firmy Quantum Software System Ltd., Kanada

Popularny (około 55000 instalacji) wieloprogramowy, wielodostępny, sieciowy system operacyjny przeznaczony dla komputerów IBM PC/XT/AT/386 i IBM PS/2 (możliwość komunikacji z dużymi komputerami m.in. IBM 360,VAX). W odróżnieniu od Unixa (Xenixa) nadaje się do zastosowań czasu rzeczywistego, zwłaszcza z komputerami Hewlett–Packard: Vectra i Nighthawk.

Jądro (około 10 kB) zawiera 28 funkcji systemowych, stanowiących jednocześnie bibliotekę języka C dla zastosowań R–T, zapewniającą:

- pełną wielozadaniowość z możliwością nadawania zadaniom priorytetów (2–15) oraz trzema trybami pracy zadań w relacji „ojciec – syn" (ang. parent – child),
- efektywną komunikację międzyzadaniową (poprzez porty, skrzynki pocztowe i tzw. wyjątki (ang. exceptions) systemowe i użytkowe,
- zadania o tym samym priorytecie są obsługiwane wg algorytmu karuzelowego (ang. round robin) z ustawionym przez specjalną funkcję systemową indywidualnym kwantem czasu (ang. time slicing),
- krótki czas reakcji systemu (maksymalny czas odpowiedzi na przerwania dla AT z zegarem 8–MHz $<300\,\mu s$, czas przełączenia zadania $<350\,\mu s$).

Obsługa przerwań składa się z handlerów (pisanych w asemblerze), komunikujących się z administratorem przerwań (będącym jednym z zadań QNX–a i pisanym w języku C) poprzez porty.

Poza językiem C zawiera kompilatory BASIC–a, FORTRAN–u i Pascala.

System PC–DOS może pracować jako jedno z zadań QNX–a.

Kompatybilny (w zakresie źródeł) z Unixem System V oraz PC–DOS–em.

Jest mało znany (w porównaniu np. z VRTX/86 lub iRMX86) w zastosowaniach czasu rzeczywistego z uwagi na stosunkowo niewielką liczbę aplikacji R–T z elementami wielodostępu i sieciowości oraz brak odpowiedniej dokumentacji.

4. Zakończenie

W pracy wymieniono kilkanaście systemów operacyjnych czasu rzeczywistego i omówiono niektóre ich charakterystyczne cechy. Warto zwrócić uwagę na dwa aspekty tych systemów.

Po pierwsze, na polskim rynku informatycznym brak jest znanych systemów występujących w świecie; dlatego też wydaje się być godnym uwagi próba wypełnienia tej luki dokonana w PIAP. Praca nad systemem SIRTOS jest warta również podkreślenia z racji tego, że system ten może służyć jako propozycja standardu minimalnego jądra.

Po drugie należy zwrócić uwagę na wpływ nowych koncepcji i technologii informatycznych na kształt tworzonego oprogramowania. Osoby projektujące systemy operacyjne czasu rzeczywistego muszą być świadome istniejących środowisk programistycznych, systemów w rodzaju Unixa czy środowiska Ady. Wybór właściwego oprogramowania narzędziowego pozwala znacznie przyśpieszyć i ułatwić pracę nad systemami czasu rzeczywistego wraz z oprogramowaniem aplikacyjnym.

Literatura

Brinch H.P.: Operating System Principles. Prentice–Hall, Englewood Cliffs, Nowy Jork 1978

DATA START, Biuro Marketingowo – Techniczne, Wrocław, [oferta handlowa systemu QNX]

Deitel H.M.: An introduction to operating systems. Addison–Wesley Publ. Comp, 1984

DeWolf J.B., Principato R.N.: A metodology for requirements specification and preliminary design of real–time systems. The Charles Stack Draper Lab. Inc, Cambridge, Mass C–4923, lipiec 1977

Evanczuk S.: Real–time operating systems – special report. Electronics, marzec 24, 1983, str. 105–115

Funck G.: Component–based operating system works in real–time. Computer Design lipiec 1984

Gallacher J.: 16–bit operating systems. Microprocessors and Microsystems, 1983 str. 364–368, Butterworth & Co (Publishers) Ltd.

Gehani N.H.: (Fault Tolerant) Concurrent C. Proceedings of the IFIP/IFAC Working Conference on Hardware and Software for Real–Time Process Control, Warszawa, maj 1988

Honeywell – „TDC 3000" oraz „TDC 2000" 1983 [opisy firmowe]

INTEL Corp. – „The 8086 Family User's Manual", 1979

ISO (1983). ISO/DP 7846. Draft Standard Industrial Real–Time FORTRAN International Standards Organization. [Również] International Purdue Workshop/ EWICS TC1, sierpień 1982

Kent Process Control Ltd.: „P4000 Distributed System", Automation Systems, Hitchin, England. [opis firmowy]

Kerninghan B.W., Mashey J.R.: The Unix programming environment. Computer, kwiecień 1981, str. 12–24

Kerninghan B.W., Ritchie D.M.: The C Programming Language. Nowy Jork Englewood Cliffs, 1978

Levene A.A.: Design techniques for real–time software systems. Proc. Eur. Conf. on Software System Eng., Londyn 1976

Lobelle M., Fanard A.: UNAC, a Distributed Real–Time UNIX for Industrial Applications. Proceedings of the IFIP/IFAC Working Conference on Hardware and Software for Real–Time Process Control, Warszawa, maj 1988

Madey J.: Ockham, Edison, Joyce i inni, czyli o programowaniu współbieżnym. Czwarta Jesienna Szkoła PTI, Mrągowo, grudzień 1987

Madnick S.E., Donovan J.J.: Operating systems. Nowy Jork, Mc Graw–Hill Book Company 1974

Martin J.T.: Programowanie maszyn cyfrowych w systemach uwarunkowanych czasowo. Warszawa, WNT 1970

Nikiel W., Wółtański S.: SIRTOS – a Small Industrial Real–Time Operating System. Proceedings of the IFIP/IFAC Working Conference on Hardware and Software for Real–Time Process Control, Warszawa, maj 1988

Partyka M.: RTMT – wielozadaniowy system operacyjny czasu rzeczywistego. Biuletyn MERA–PIAP 3/116, 1986

Perycz K.: Mikrokomputerowy system operacyjny SI. Informatyka 5 i 6/1986

Pettersen O.: ARTEX – A real–time executive for distributed process control. WRTP'86 IFIP's Workshop, Lake Balaton, 1986

Mikrokomputery szesnastobitowe – systemy operacyjne. Polskie Towarzystwo Informatyczne. Materiały III Szkoły, Łódź, grudzień 1986

Strzałkowski P.: Conception d'un Language temps reel sur la base du language APL. SES/INTERNE/SIR/78–198, CEA–CENS, Services d'Electronique de Saclay, 1978

Strzałkowski P.: Systemy ekspertowe w procesach technologicznych. Informatyka 12/1987

Technical Reference IBM Personal Computer, 1984

TECSI – „REALTIME CRAFT. A real–time multitasking operating system for the main 16–32 bit microprocessors". 1986 [opis firmowy]

Using PSL/PSA (version A4.2) for Real–Time Systems ISDOS Project Ann Arbor, Michigan, ISDOS Ref. 79000–0249–0

Valmet Corp. – Damatic automation system. [opis firmowy]