Wykład 10

Uruchamianie i testowanie systemów mikroprocesorowych

Bartłomiej Zieliński, PhD, DSc

Uruchamianie i testowanie

Program:

- Problemy podczas uruchamiania systemów µp
  - Problemy sprzętowe
  - Problemy programowe
  - Problemy we-wy
- Analiza systemów µp
  - Analiza asynchroniczna
  - Analiza synchroniczna
- Narzędzia uruchomieniowe
  - Debugery, symulatory
  - Analizatory stanów logicznych
  - Emulatory układowe

Uruchamianie i testowanie

- Systemy µp
  - µp są VLSI
    - Brak dostępności sygnałów wewnętrznych

      →obserwacja magistral zewnętrznych
  - µp pracuje zgodnie z programem
    - Wyniki działania sprzętu i oprogramowania

      →dane zbierane jednocześnie z wszystkich magistral

- Uruchamianie i testowanie systemów µp
  - Nowe metody i narzędzia

Uruchamianie i testowanie

- Problemy podczas uruchamiania systemów µp
  - Problemy sprzętowe
    - „zimny lut” („*cold solder*”), zły kontakt
      → kontakt elektryczny słaby, zły lub brak kontaktu
    - Uszkodzenie lub zniszczenie układu scalonego
    - Błędny montaż układu w płytce
    - Zły kontakt układu scalonego z podstawką
    - Błąd projektu płytki
      → przesłuchy, zakłócenia
    - Wysoka oporność ścieżek zasilania i masy
      → niewłaściwe poziomy napięć sygnałów logicznych
      → zmniejszenie marginesu zakłóceń

Uruchamianie i testowanie

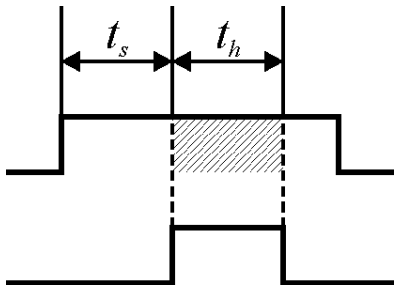
- Problemy podczas uruchamiania systemów µp
  - Problemy programowe
    - Sprzęt OK, program OK, ale współpracują źle
      - Praca krokowa
    - Zapętlanie programu – skok w losowe miejsce
    - „wykonywanie danych”
    - Czasem działa, czasem nie
      - „*Jeśli coś działało i przestało, to coś musiało się zmienić*” (Alex Ragen: „Leksykon języka C” (?))
  - Problemy we-wy
    - Błędna prędkość transmisji
    - Błędny format danych
    - Zakłócenia linii przesyłowych

Uruchamianie i testowanie

- Analiza synchroniczna
  - Zbieranie cykli maszynowych zgodnie z zegarem µp
    - Wyzwalanie określonym słowem
      - Obserwacja tylko fragmentu programu
      - Zapamiętanie także słów poprzedzających wyzwolenie
        → *jak µp dotarł do tego miejsca programu?*
        - Skok / następny rozkaz / błąd
    - Zliczanie wystąpienia słowa wyzwalającego
      - Gdy układ działa lub nie
      - Analiza rozpoczyna się od *n*-tego przejścia

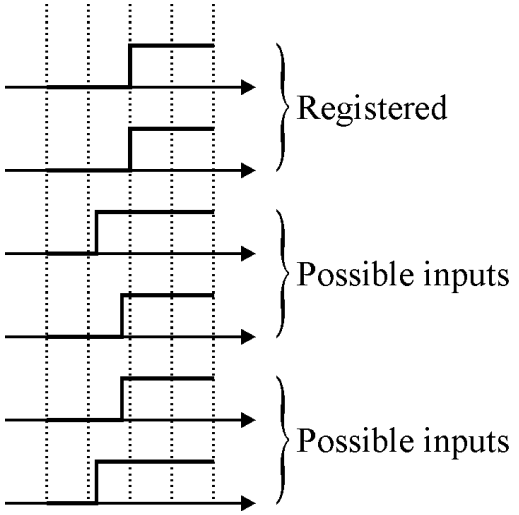
Uruchamianie i testowanie

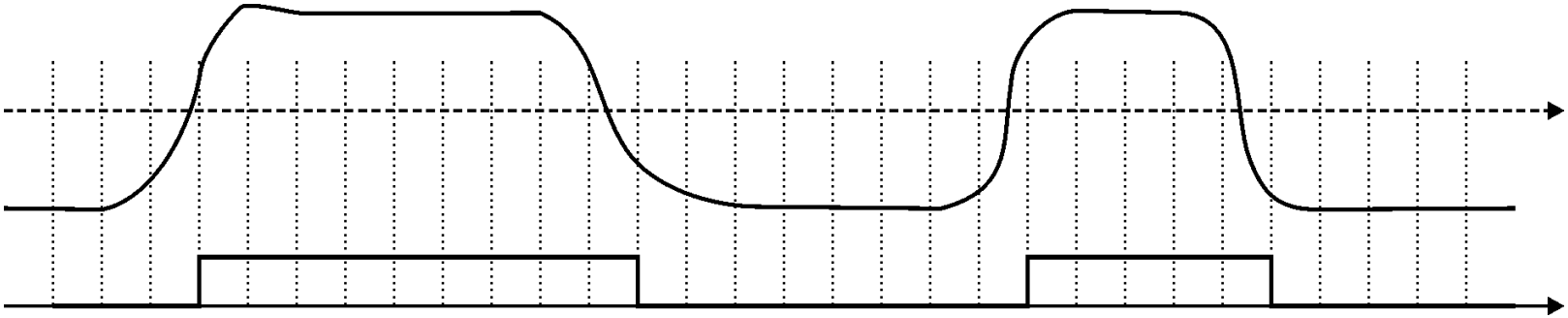
- Analiza synchroniczna
  - Zbieranie cykli maszynowych zgodnie z zegarem µp
    - Wyzwolenie „opóźnieniem logicznym”
      - Gdy nie można określić słowa wyzwalającego
        - » Wystąpienie słowa („ostatnie dobre”)
        - » Odliczanie *n* cykli
        - » Rozpoczęcie rejestracji
    - Nie zawsze uwzględnione parametry dynamiczne
      - Czas ustalania t_s
      - Czas podtrzymania t_h



Uruchamianie i testowanie

- Analiza asynchroniczna
  - Próbkowanie zgodnie z zegarem analizatora
    - Bardzo wysoka częstotliwość
    - Próbkowanie wszystkich kanałów
  - Niejednoznaczność wyników
    - $f \neq \infty$
    - Wewnętrzne opóźnienia analizatora



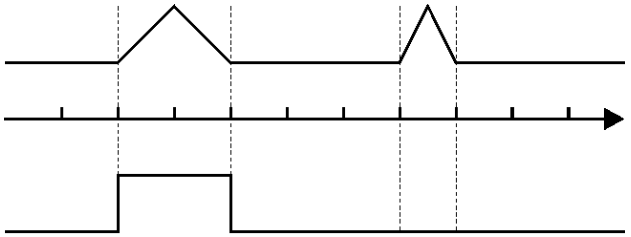


Uruchamianie i testowanie

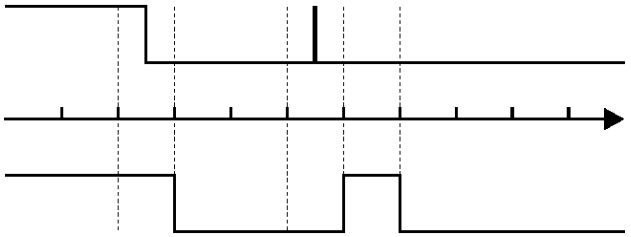
- Analiza asynchroniczna
  - Wykrywanie impulsów szpilkowych (*glitch*)
    - *Co to jest „szpilka”?*
      - Bardzo krótki impuls
      - Przyczyna: przesłuchy, zakłócenia, błędne Vcc/Gnd
      - Skutek: fałszywe przerwania, błędy zliczania, błędy wymiany danych itp.
    - *Jak wykryć „szpilkę”?*
      - Bardzo wysoka częstotliwość (np. 200 MHz → 5 ns szpilka)
        - » (szpilka < 5 ns nie bardzo ważna dla µp)
      - Przerzutniki zatrzaskowe
        - » *Czy był ciąg szpilek, czy tylko jedna?*
      - Oddzielne zapisywanie sygnału i szpilek

Uruchamianie i testowanie

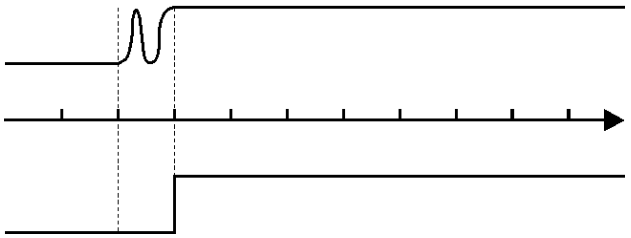
- Analiza asynchroniczna
  - Wykrywanie szpilek
    - Próbkowanie



    - z rejestrami zatrzaskowymi

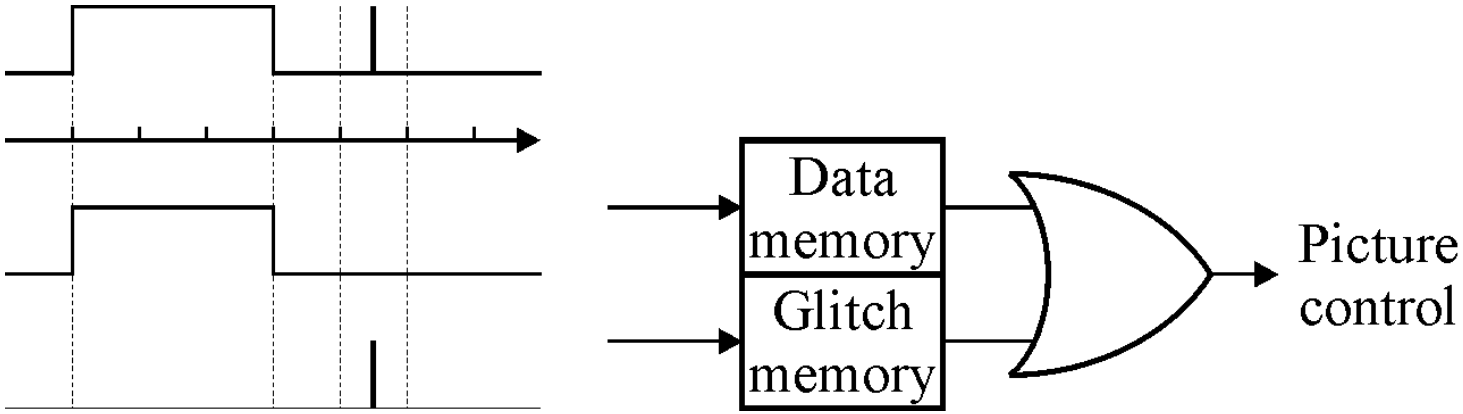


    - Szpilki mogą być pominięte przez wejścia analizatora



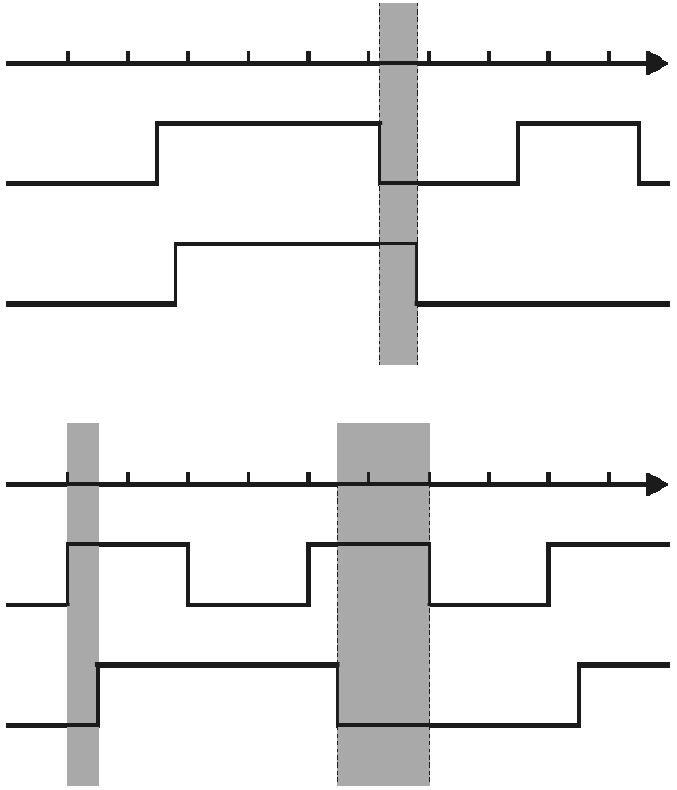
Uruchamianie i testowanie

- Analiza asynchroniczna
  - Wykrywanie szpilek
    - Oddzielne zapisywanie danych i szpilek



Uruchamianie i testowanie

- Analiza asynchroniczna
  - Niejednoznaczność warunków wyzwalania
    - Pominięte wyzwolenie
      (między próbkami)
    - Fałszywe wyzwolenie
      (np. różnice opóźnień)



Uruchamianie i testowanie

- Analiza asynchroniczna
  - Niejednoznaczność warunków wyzwalania
    - Zależności czasowe sygnałów względem zegara
      - opóźnienie = czas propagacji układu i porównanie warunku wyzwolenia dopiero potem
    - Zapis z opóźnieniem względem wyzwolenia
      - Interesujące skutki nieciekawego zdarzenia
    - Wyzwalanie szpilkami
      1. Zapis 1 słowa
      2. Brak szpilek → powtórz 1
      3. Szpilki → rejestracja przez dłuższy czas

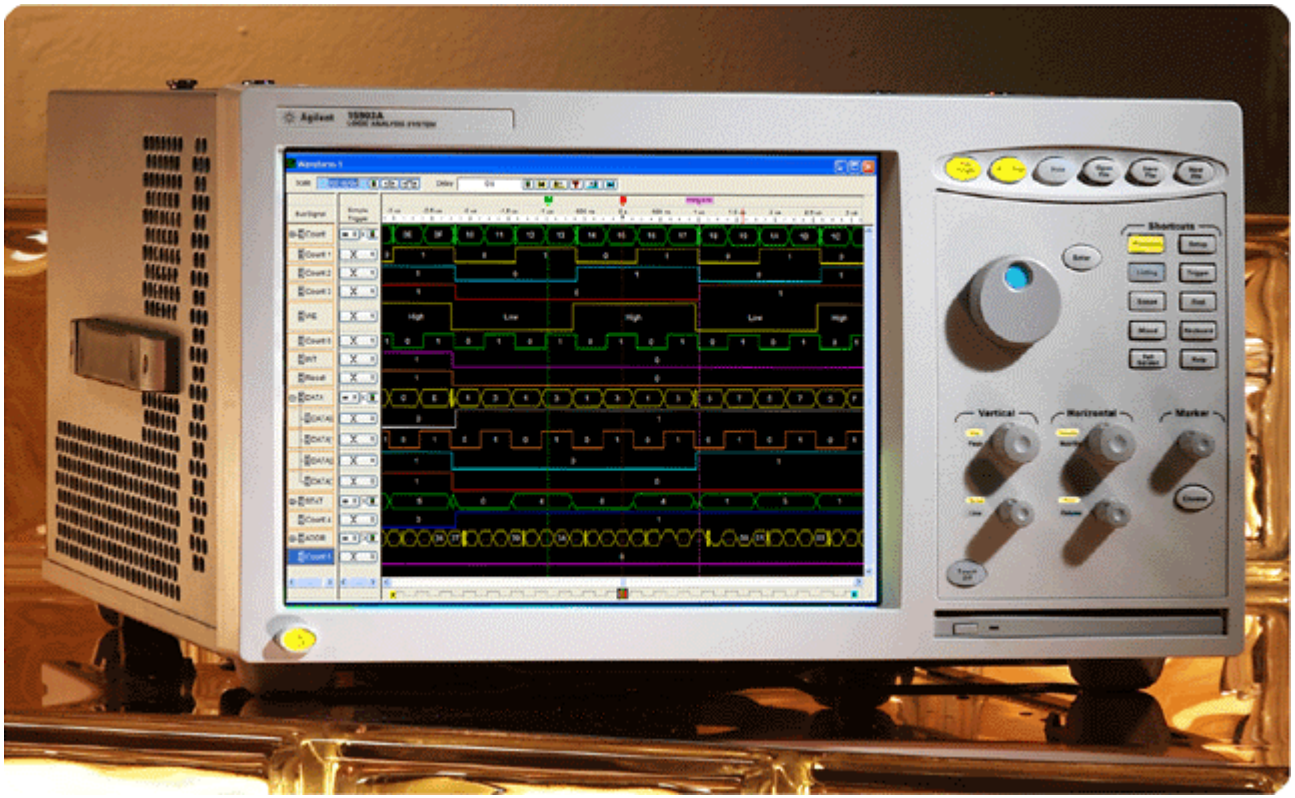
      → kiedy wystąpiła szpilka i jakie były jej skutki

Uruchamianie i testowanie

- Analiza – częste problemy
  - Niewłaściwe zasilanie, błąd montażu układu
  - Błędne poziomy napięć (np. TTL – CMOS)
  - Niepewne połączenia
    - Analizator pokazuje inne stany, niż widzi µp
  - Czas ustawiania, czas podtrzymania
  - itp.

Uruchamianie i testowanie

- Analizator stanów logicznych



https://vishnuanirudh.files.wordpress.com/2012/07/logic-analyzer.gif

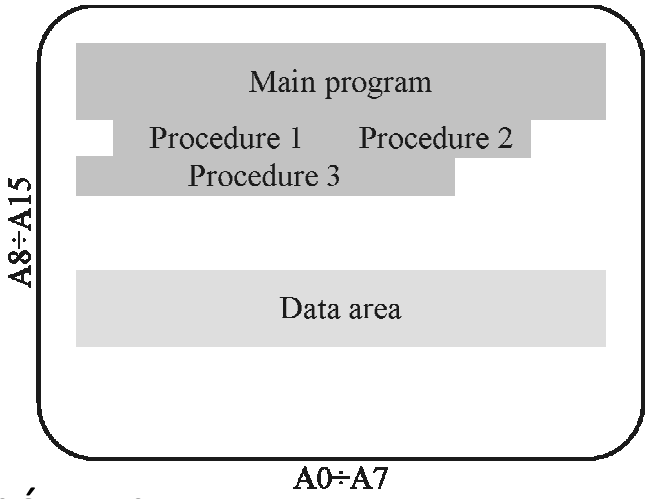
Uruchamianie i testowanie

- Analizator stanów logicznych

https://eleshop.eu/sla1016.html

Uruchamianie i testowanie

- Analizator stanów logicznych – funkcje
  - Jak oscyloskop, ale:
    - Dużo jednocześnie obserwowanych wejść (np. > 100)
    - Sygnały cyfrowe
    - Wyzwalanie wzrorcem
  - Wyjście:
    - Ciągi „0” i „1”
    - Przebiegi czasowe wielu sygnałów
    - Mapa dostępu do pamięci
      - Użyteczne przy uruchamianiu systemów µp
    - Przebieg programu (deasemblowany)

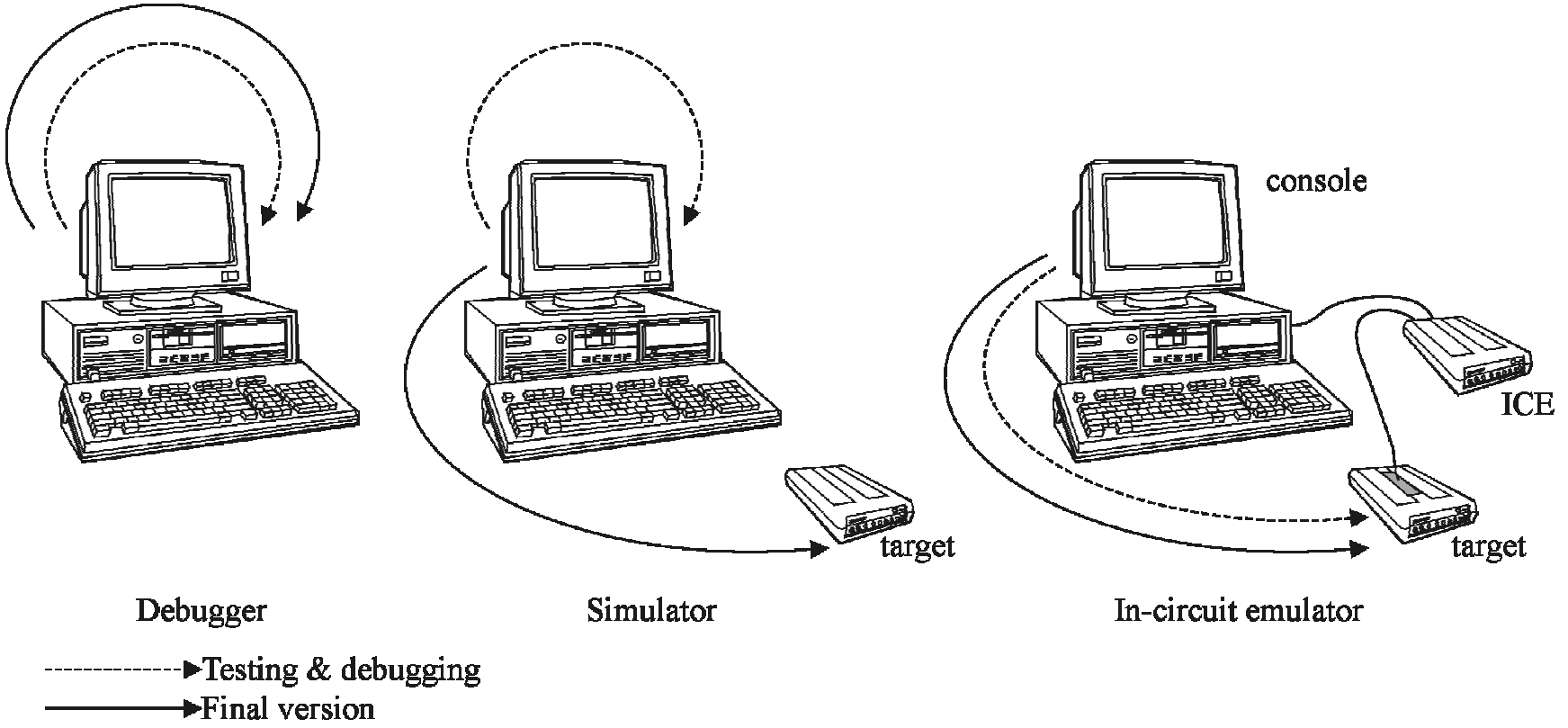


Uruchamianie i testowanie

- Analizator stanów logicznych – podsumowanie
  - Skuteczne narzędzie przy znajdowaniu błędów sprzętowych i programowych
  - Tylko obserwacja układu
    - Pasywna rejestracja zachowania układu
    - Brak wpływu na układ
    - Można odnaleźć błędy, ale nie można ich naprawić
  - *Przydałyby się inne narzędzia*

Uruchamianie i testowanie

- Debuger / symulator / emulator układowy

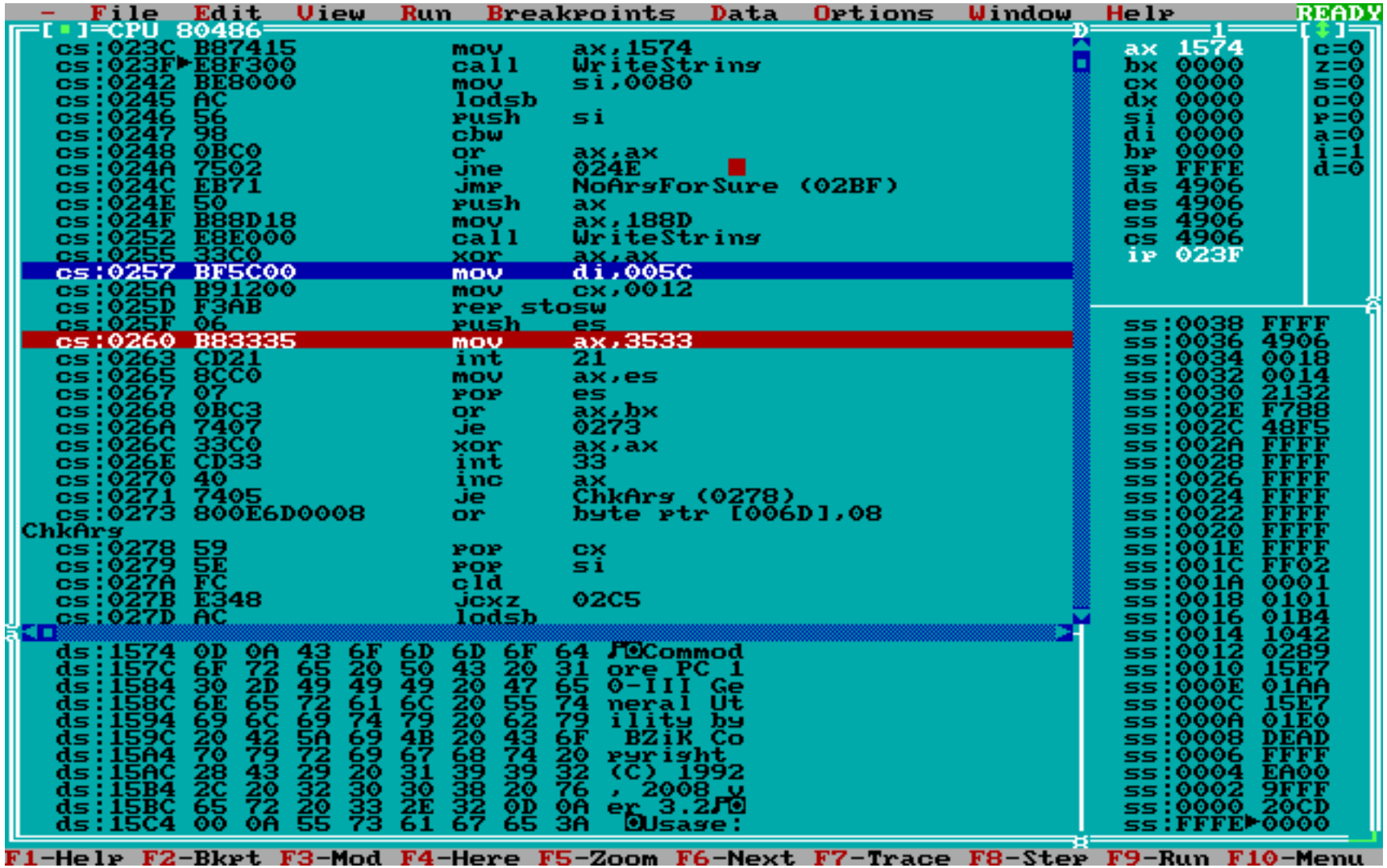


Uruchamianie i testowanie

- Debuger i symulator – funkcje
  - Wykonanie programu ciągłe i krokowe
    - Niektóre symulatory umożliwiają „krok wstecz”
  - Wyświetlanie programu jako kodu asemblerowego
    - Możliwość użycia nazw symbolicznych
    - Możliwe uruchamianie na poziomie instrukcji języka wysokiego poziomu
      - Wymagana informacja o symbolach z kompilatora i linkera
  - Wyświetlanie i modyfikacja zawartości rejestrów µp
  - Wyświetlanie i modyfikacja zawartości pamięci

Uruchamianie i testowanie

- Przykład – Turbo Debugger 2.0

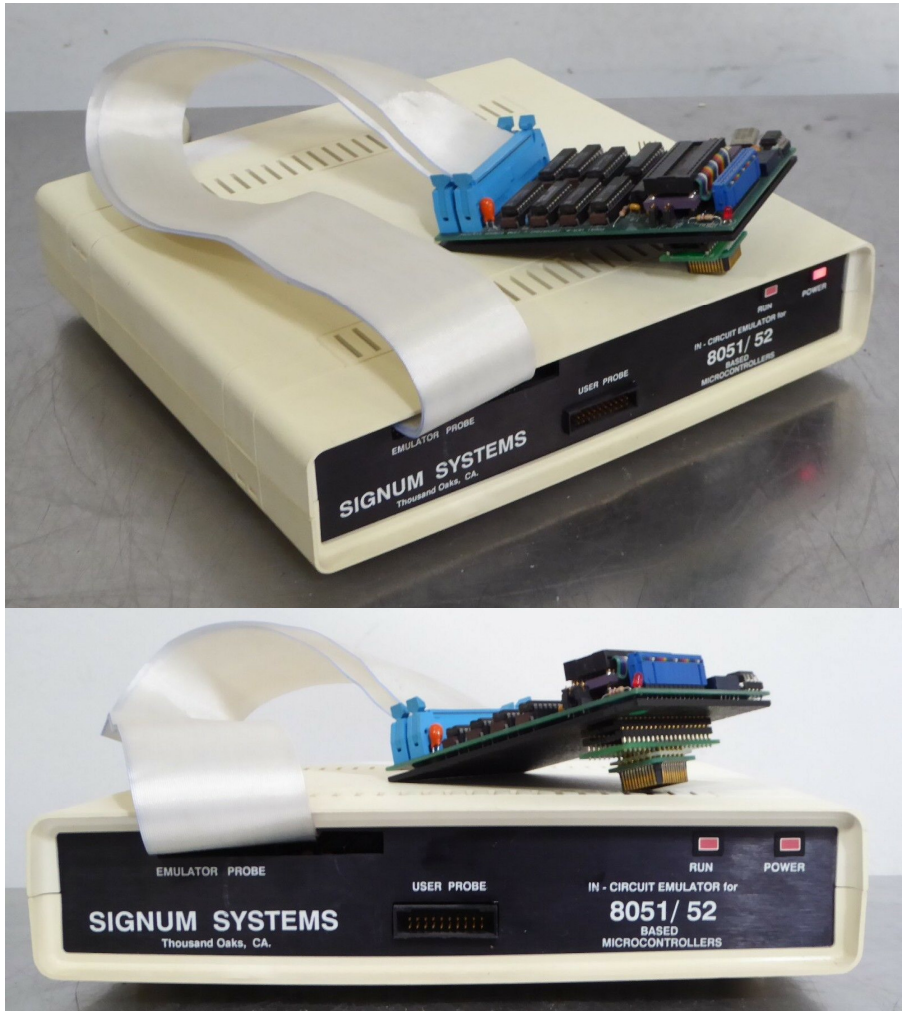


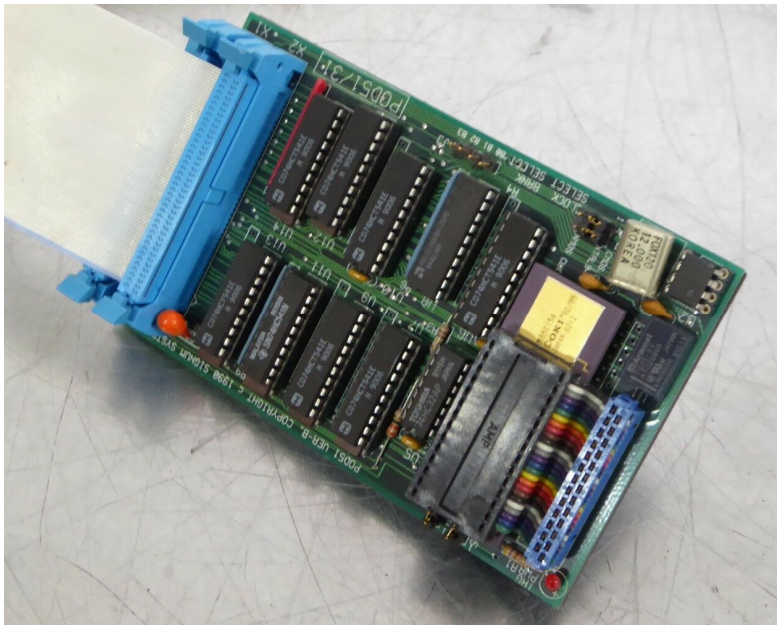
Uruchamianie i testowanie

- Symulator a debuger
  - Symulator działa w środowisku „wirtualnym”
    - Rozkazy → procedury
    - Rejestry → zmienne
    - Pamięci układu docelowego → tabele
    - We-wy układu docelowego → rekordy + procedury
    - Konieczne określenie zakresów adresów pamięci i we-wy
  - Wymagania symulacji w czasie rzeczywistym:
    - Szczegółowa wiedza o cyklach pracy µp (cykle rozkazowe, maszynowe, zegarowe)
    - Znacząco wyższa moc obliczeniowa (np. Pentium II 300 MHz może symulować 8051 12 MHz)

Uruchamianie i testowanie

- Emulator układowy – przykład
  - Signum Systems ICE-51



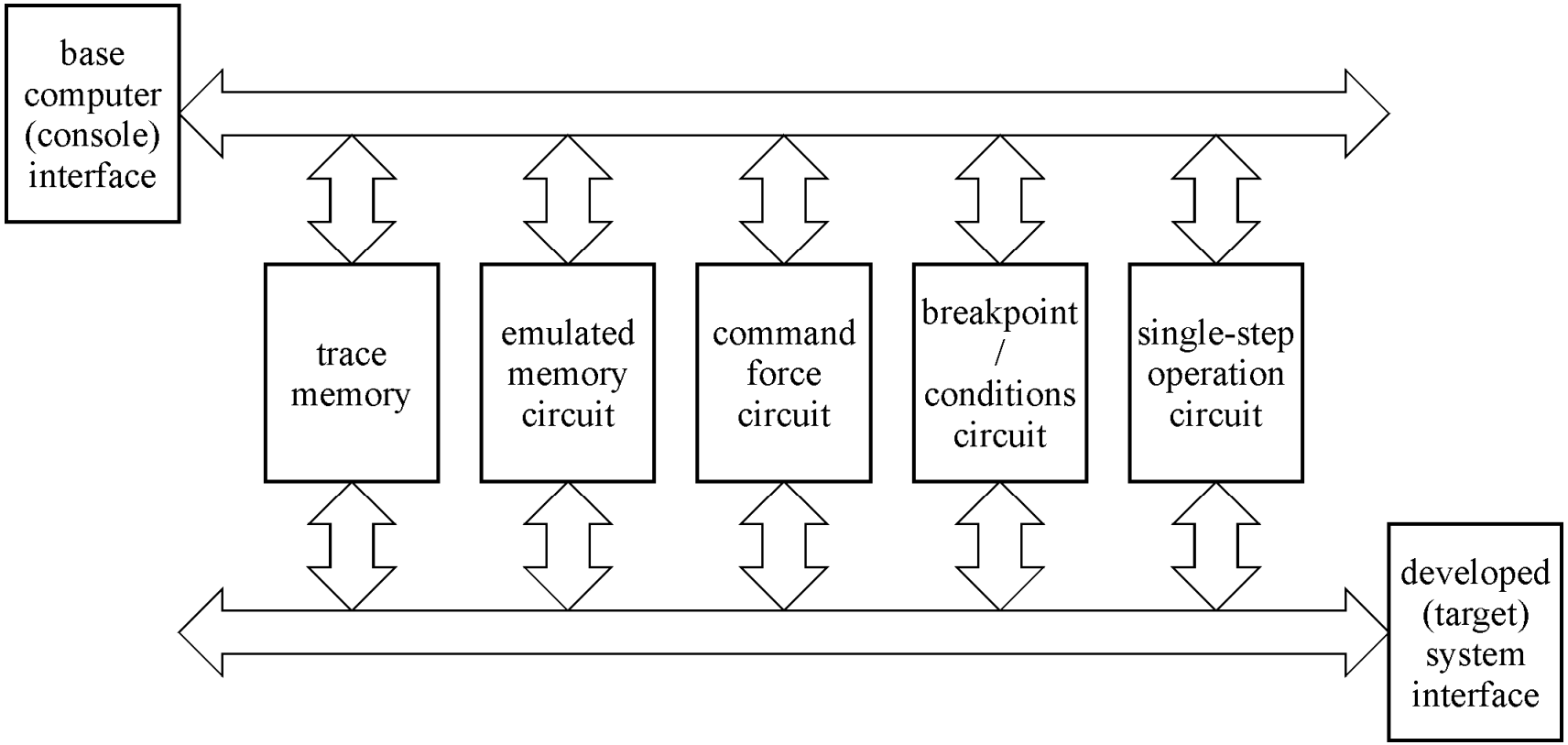


Uruchamianie i testowanie

- Emulator układowy
  - Użytkownik „widzi” system dokładnie tak, jak µp
    - Typy i adresy pamięci
    - Układy we-wy
    - Otoczenie zewnętrzne
  - Można znaleźć wszystkie niedoskonałości
    - Błędy logiczne projektu (np. błąd dekodowania adresów)
    - Błędy projektu i wykonania płytki
    - Błędy montażu układów na płytce
    - Problemy integracji sprzętu i oprogramowania
    - Problemy czysto programowe

Uruchamianie i testowanie

- Emulator układowy – przykładowa struktura



Uruchamianie i testowanie

- Emulator układowy – przykładowa struktura
  - Układ pracy krokowej
    - Tryby pracy
      - Praca ciągła
      - Zatrzymanie na najbliższym rozkazie
        - » Uruchamianie/testowanie oprogramowania
      - Zatrzymanie na najbliższym cyklu maszynowym
        - » Uruchamianie/testowanie sprzętu
    - Jak zaimplementować?
      - „Klasyczne” µp (np. Z80, 8086 itp.) – wejście wydłużenia cyklu maszynowego (np. $\overline{\mathrm{Wait}}$, Ready itp.)
      - Jednoukładowe / µs – musi mieć wbudowany mechanizm (np. 8048 ma wejście $\overline{SS}$, 8051 – brak wsparcia)

Uruchamianie i testowanie

- Emulator układowy – przykładowa struktura
  - Układ pułapek
    - Prosty
      - Adres i/lub dane równe podanej wartości
      - Wystąpienie określonego cyklu maszynowego
    - (Nie tak) prosty
      - Zgodność adresu i/lubn danych z wzorcem
        - » Porównanie tylko wybranych bitów
      - Koniunkcja prostych warunków

Uruchamianie i testowanie

- Emulator układowy – przykładowa struktura
  - Układ pułapek (warunków)
    - Złożony (np. Signum Systems ICE-51)
      - Zdarzenia
        - » Wybrany cykl maszynowy (+zgodność adresu, danych)
        - » Sygnały użytkownika
        - » Zliczanie zdarzeń
        - » Kolejność zdarzeń
        - » Zapełnienie pamięci śladu
      - Skutki
        - » Pułapka
        - » Start/stop licznika zdarzeń
        - » Koniec śladowania

Uruchamianie i testowanie

- Emulator układowy – przykładowa struktura
  - Układ forsowania rozkazów
    - µp wykonuje rozkaz lub procedurę, których nie ma w pamięci programu
    - *... ale do czego nam to potrzebne?*
      - Szybka „ręczna” naprawa błędu bez rekompilacji programu
      - Wykonanie procedury diagnostycznej
        - » Np. zapisanie zawartości rejestru do pamięci po napotkaniu pułapki
        - » Aktualizacja okna stanu µp
      - Szybkie sprawdzenie określonych działań przed ich implementacją w programie/w systemie

Uruchamianie i testowanie

- Emulator układowy – przykładowa struktura
  - Układ pamięci emulowanej
    - Zastępuje lub uzupełnia systemową pamięć RAM/ROM
    - Zawiera tylko pamięć RAM
      - Wyższy priorytet niż pamięć w systemie
        - » „ukrywa” (zastępuje) pamięć systemową
      - Możliwość łatwej modyfikacji
        - » Szczególnie, gdy pamięć systemowa to E(P(ROM))
      - Możliwość określenia zakresu adresów
    - *A co z układami we-wy?*

Uruchamianie i testowanie

- Emulator układowy – przykładowa struktura
  - Układ (pamięci) śladowania
    - Zapisuje ślad wykonania programu
      - Wszystkie cykle maszynowe na wyprowadzeniach µp
    - Można zapamiętać także sygnały użytkownika
    - Wyświetlenie śladu – przepływ programu ze zdekodowanymi rozkazami, danymi, sygnałami we-wy…
      - Jak analizator stanów logicznych (analiza synchroniczna)

Uruchamianie i testowanie

- JTAG – współczesne metody
  - *Joint Test Action Group* (1985)
    - IEEE Std. 1149 (1990)
  - Zastosowania pierwotne
    - Testowanie i diagnozowanie urządzeń, płyt i systemów
  - Zastosowania współczesne
    - Dostęp do pod-bloków układu
    - Uruchamianie systemów wbudowanych
    - Programowanie (oprogramowanie układowe, *firmware*)
    - Skanowanie peryferyjne (*boundary scan*)

Uruchamianie i testowanie

- JTAG – interfejs
  - Kolejny interfejs wbudowany w µp/µs
  - „pełny” JTAG
    - TDI (*Test Data In*)
    - TDO (*Test Data Out*)
    - TCK (*Test Clock*)
    - TMS (*Test Mode Select*)
    - TRST (*Test Reset*, optional)
  - „zwarty” JTAG – cJTAG (*compact JTAG*)
    - TMSC (*Test Serial Data*)
    - TCKC (*Test Clock*)