

Procesorski sistemi v telekomunikacijah
Nalaganje in razhroščevanje programov

(c) Arpad Bűrmen, 2010-2012

Zapis programa s programatorjem

- ▶ Program v (E)EPROM/FLASH pomnilniku v ločenem čipu
- ▶ Čip odstranimo iz sistema in vanj s pomočjo programatorja zapišemo program
- ▶ Običajno je treba program pretvoriti iz oblike, ki jo dobimo iz povezovalnika v obliko, ki jo pozna programator
- ▶ Formati: Intel HEX, Motorola S (S19), ... od leta 1970 naprej
- ▶ Pogosto je FLASH pomnilnik v istem čipu kot procesor...
Procesor vstavimo v programat
in vanj zapišemo program

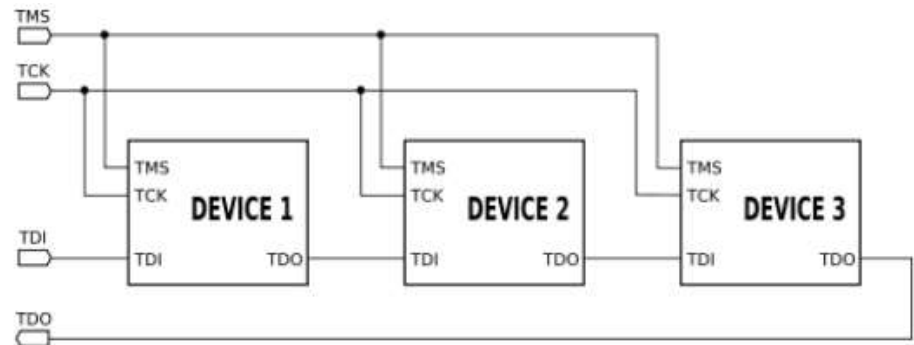


Programiranje med delovanjem (In System Programming, ISP)

- ▶ Nekateri mikroprocesorji z vgrajenim FLASH pomnilnikom omogočajo, da v njih prenesemo program, brez da bi jih odstranili iz vezja
(In System Programming, ISP)
- ▶ Najbolj pogosta načina nalaganja programa sta serijska povezava in vmesnik JTAG
- ▶ Primer: v LPC2138 lahko po reset-u (če je P0.14 na nizkem nivoju) naložimo program preko vmesnika UART0 (P0.0, P0.1 = TxD0, RxD90) v FLASH pomnilnik s hitrostmi do 230400b/s (23kB/s)

Vmesnik JTAG

- ▶ JTAG (Joint Test Action Group, IEEE 1149.1)
- ▶ Podoben vodilu SPI, 4 signali:
 - TMS (test mode select) – enakovreden SS (slave select) signalu SPI
 - TCK (test clock) – enakovreden SCLK signalu SPI
 - TDI (test data input) – enakovreden MOSI signalu SPI
 - TDO (test data output) – enakovreden MISO signalu SPI
- ▶ Verižna vezava naprav, ki jih krmilimo preko JTAG povezave
Pogosto imamo le eno napravo v verigi, t.j. mikroprocesor
- ▶ Omogoča krmiljenje in odčitavanje stanja naprav v verigi.
- ▶ Tipične hitrosti od 10Mb/s do 100Mb/s
- ▶ Primer: JTAG v LPC2138 P1.27-P1.30 ... TDO, TDI, TCK, TMS
- ▶ Standard ne določa ukazov, ki se prenašajo preko povezave JTAG
Vsak proizvajalec ima svoje ukaz



Razhroščevanje (debugging)

- ▶ Postopek iskanja napak v programu.
- ▶ **Preprost pristop:** takoimenovano **printf razhroščevanje**
V program dodamo stavke printf, ki redno izpisujejo vrednosti spremenljiv in stanje programa. Izpisana sporočila beremo preko serijske povezave ali pa na prikazovalniku (pogosto je ta del mikroprocesorskega sistema).
- ▶ V skrajnih primerih namesto tekstovnih sporočil uporabljamo lučke ali pa kar signale, ki jih opazujemo na osciloskopu.
- ▶ Pristop je preveč preprost, da bi z njim lahko odkrili večje napake.
- ▶ **Posebni programi in strojna oprema za razhroščevanje – razhroščevalnik (debugger)**
Z njihovo pomočjo lahko
 - ustavimo program na določenih mestih (prekinitvene točke, breakpoints),
 - gremo skozi program po korakih (korakanje, stepping),
 - pregledujemo vrednosti spremenljivk, registrov in pomnilnika
- ▶ ⁵(watch)

Razhroščevalnik (Debugger)

Primer: GDB

```
File Edit View Terminal Tabs Help
john@anaconda:~/Desktop/cpp$ gdb sample
GNU gdb 6.8-debian
Copyright (C) 2008 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu"...

(breakpoint 1) b main
Breakpoint 1 at 0x80485a5: file sample.cpp, line 24.

(breakpoint 1) f
Starting program: /home/john/Desktop/cpp/sample
[Thread debugging using libthread_db enabled]
[New Thread 0xb744e946 (LWP 6755)]
[Switching to Thread 0xb744e946 (LWP 6755)]

Breakpoint 1, main () at sample.cpp:24
24   for(t=0; t<NUM_THREADS; t++){

(breakpoint 1) list
19   pthread_t threads[NUM_THREADS];
20
21
22   int rc;
23   long t;
24   for(t=0; t<NUM_THREADS; t++){
25     printf("In main: creating thread %ld\n", t);
26     rc = pthread_create(&threads[t], NULL, PrintHello, (void *)t);
27
28

(breakpoint 1) list
29   if (rc){
30     printf("ERROR: return code from pthread_create() is %d\n", rc);
31     exit(-1);
32   }
33 }
34 pthread_exit(NULL);
35 }
```

Konzolni način dela v GDB

The screenshot shows the DDD (Data Display Debugger) GUI for GDB. The title bar indicates the file path: `DDD: /home/bitwalk/mingw-cross/work/ex_gdb/sample.c`. The interface includes a menu bar (File, Edit, View, Program, Commands, Status, Source, Data, Help) and a toolbar with icons for various actions like LookUp, Find, Break, Watch, Print, Display, Plot, Hide, Rotate, Set, and Undisp. The main window is divided into several panes:

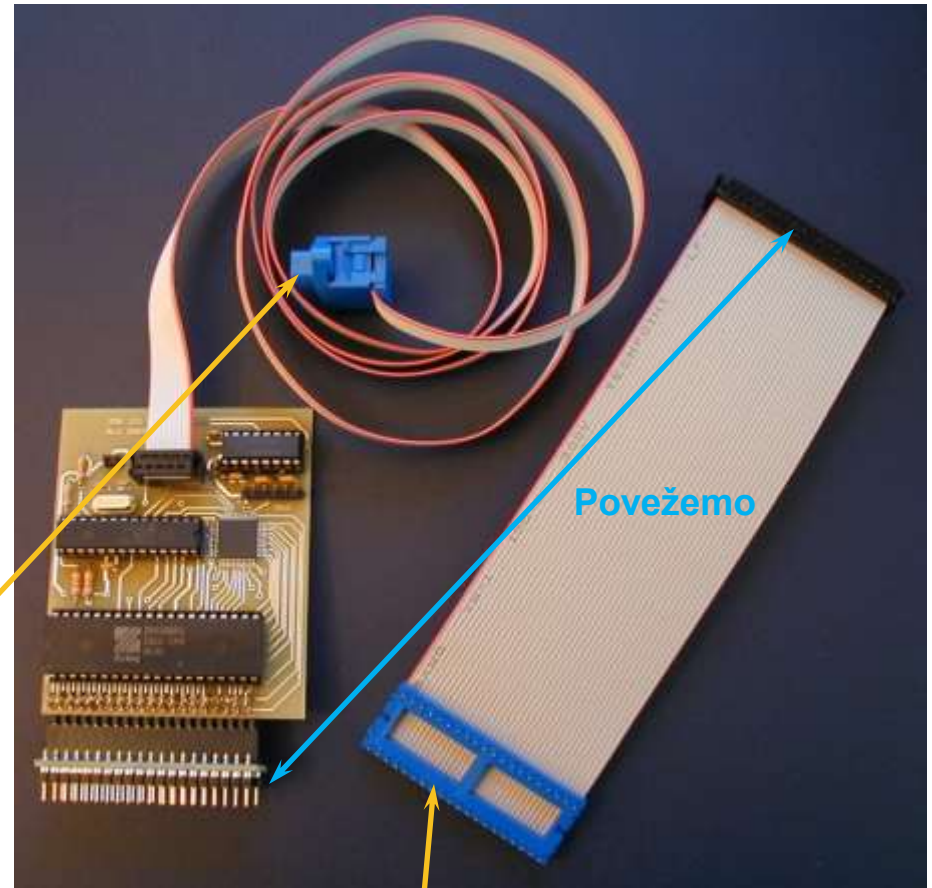
- Registers:** Shows the state of registers `f1`, `f2`, and `i`. `f1` contains the value `135.79100324316485`, `f2` contains `0`, and `i` contains `69`.
- Source Code:** Displays the C source code for `sample.c`. The current line of execution is line 32: `f1 += func(x1 + i * h);`. Red stop signs indicate breakpoints at lines 32 and 33.
- Assembly:** Shows the assembly instructions corresponding to the source code. The current instruction is `0x004013b8 <main+98>: jmp 0x4013d7 <main+129>`. Other instructions include `movl $0x1,-0x50(%ebp)`, `fildl -0x50(%ebp)`, `fmull -0x48(%ebp)`, and `faddl -0x38(%ebp)`.
- Breakpoints:** Shows the status of breakpoints. The current breakpoint is at line 32: `Breakpoint 2, main () at sample.c:32 (gdb) cont`.

Grafični uporabniški vmesnik (GUI) za GDB

Strojna podpora za razhroščevanje

Emulatorji

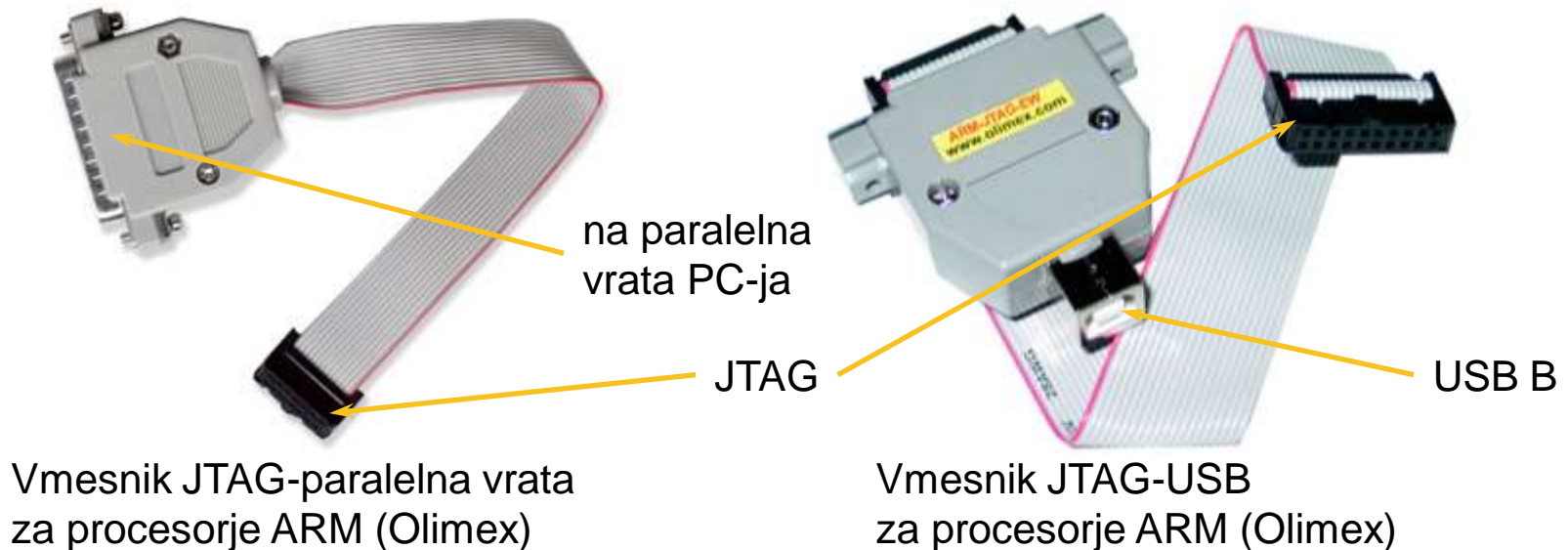
- ▶ Emulator (In Circuit Emulator, ICE) je vezje, ki nadomesti mikroprocesor.
- ▶ Vstavimo ga v mikroprocesorski sistem namesto mikroprocesorja.
- ▶ Hkrati ICE priklopimo še na osebni računalnik, kjer s pomočjo ustrezne programske opreme (pripadajočega razhroščevalnika) iščemo napake v programu.
- ▶ Na sliki je emulator Z80 – Z80 ICE.
K osebni računalniku (RS-232)
- ▶ Relativno visoka cena.



V ciljni sistem namesto Z80

Mikroprocesorji z vgrajenim emulatorjem

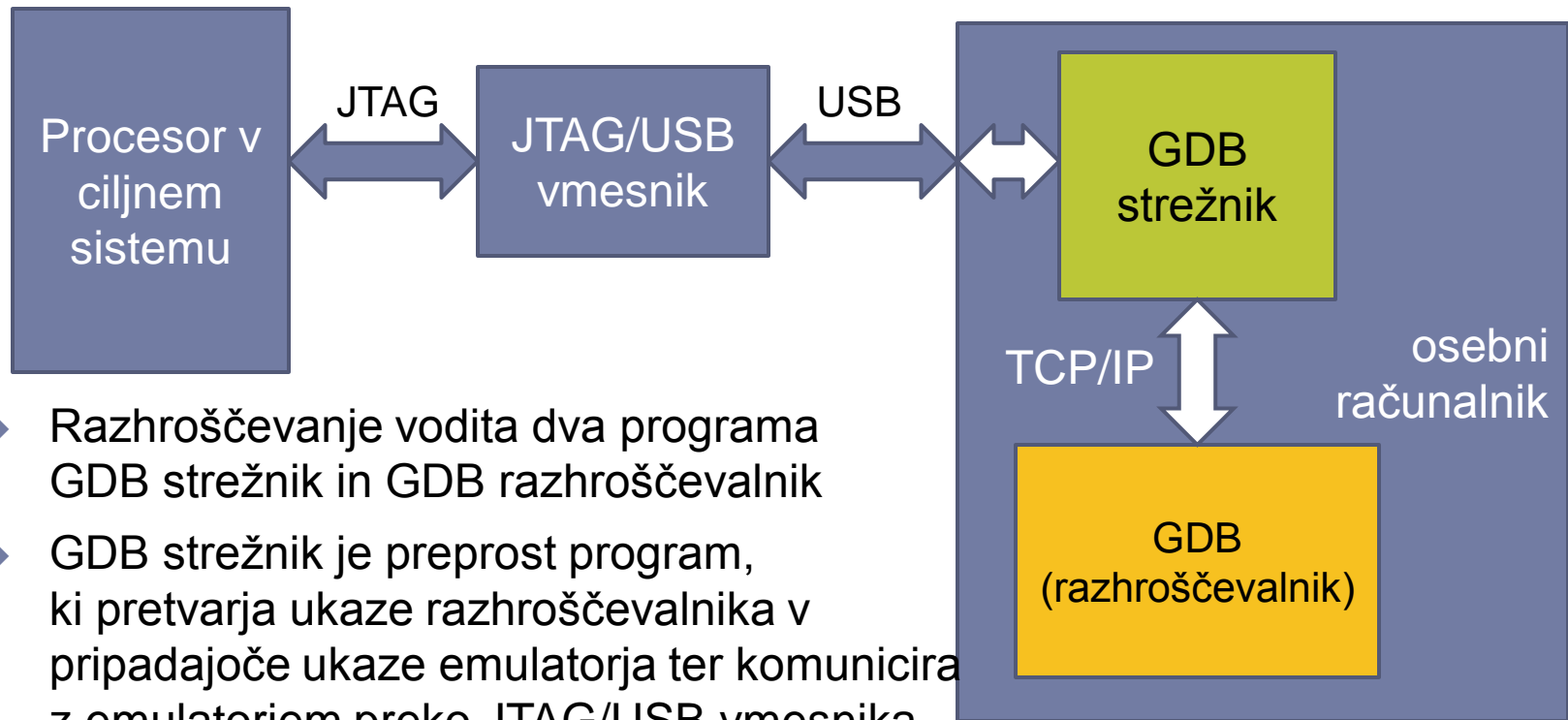
- ▶ Večina današnjih mikroprocesorjev.
- ▶ Komunikacija z emulatorjem poteka ponavadi preko vmesnika JTAG.
- ▶ Za povezavo z osebnim računalnikom skrbi vmesniško vezje.
- ▶ Programska oprema za razhroščevanje mora znati komunicirati z JTAG vmesnikom in preko njega z emulatorjem v mikroprocesorju.
- ▶ Pri okolju WinIDEA (š-ARM) je vmesnik USB/JTAG vgrajen v ploščico š-ARM.



Priprava na razhroščevanje

- ▶ Program je potrebno prevesti v posebno obliko za razhroščevanje, ki vsebuje pomožne informacije (debug information) o spremenljivkah in stavkih (npr. kje v pomnilniku se hranijo).
- ▶ Pomožne informacije so lahko del objektnih datotek in končnega programa (npr. pri gcc) ali pa se hranijo v ločenih datotekah (npr. Microsoftov prevajalnik za C).
- ▶ Program je treba zagnati s pomočjo razhroščevalnika, ki takoj po zagonu prevzame nadzor nad mikroprocesorjem.
- ▶ Razhroščevalnik ponavadi teče na osebnem računalniku, ki je preko vodila USB in vmesnika USB/ITAG povezan s cilinim mikroprocesorjem, ki

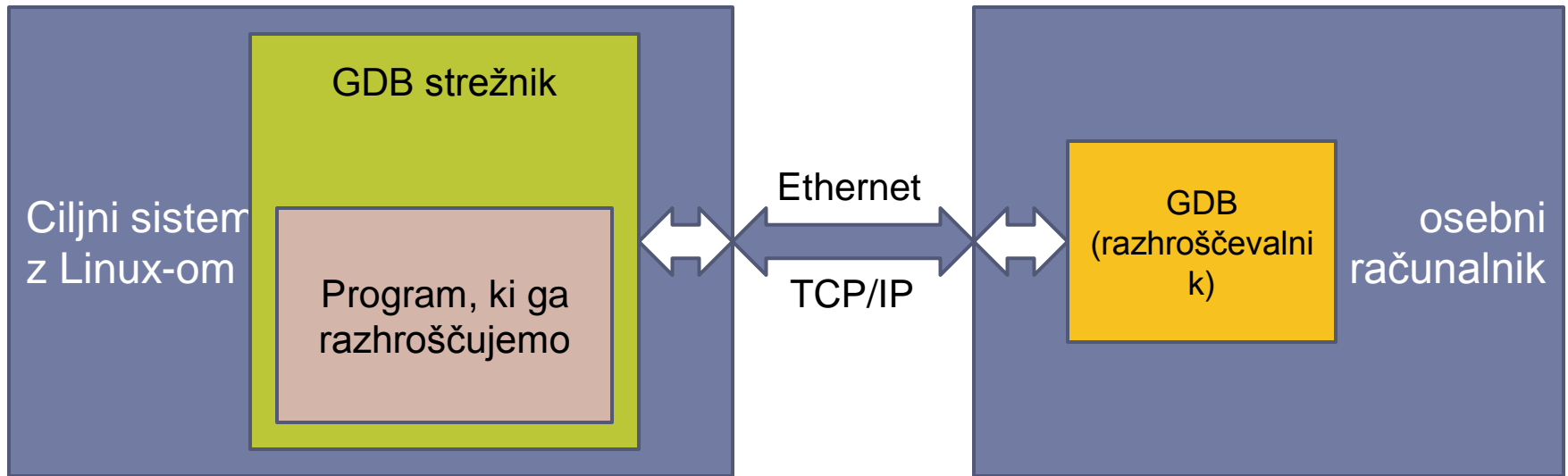
Zgradba sistema za razhroščevanje na osnovi GNU orodij gcc/gdb



- ▶ Razhroščevanje vodita dva programa GDB strežnik in GDB razhroščevalnik
- ▶ GDB strežnik je preprost program, ki pretvarja ukaze razhroščevalnika v pripadajoče ukaze emulatorja ter komunicira z emulatorjem preko JTAG/USB vmesnika
- ▶ TCP/IP komunikacija med GDB strežnikom in GDB razhroščevalnikom.
- ▶ Uporabnik (programer) upravlja razhroščevanje preko GDB razhroščevalnika.
- ▶ TCP/IP omogoča, da GDB strežnik in GDB razhroščevalnik tečeta na dveh ločenih računalnikih, ki sta povezana preko interneta.

Ciljni sistemi z operacijskim sistemom

Primer: Linux/gcc/gdb



- ▶ Program mora biti naložen na ciljni sistem, zaženemo GDB strežnik.
- ▶ Ob zagonu mu podamo program, ki ga bomo razhroščevali.
- ▶ Na osebнем računalniku zaženemo GDB razhroščevalnik.
- ▶ GDB razhroščevalnik se poveže z GDB strežnikom preko mrežne povezave.
- ▶ TCP/IP komunikacija med GDB strežnikom in GDB razhroščevalnikom.
- ▶ Razhroščevanje programa poteka s podporo operacijskega sistema. JTAG vmesnik in emulator nista potrebna.

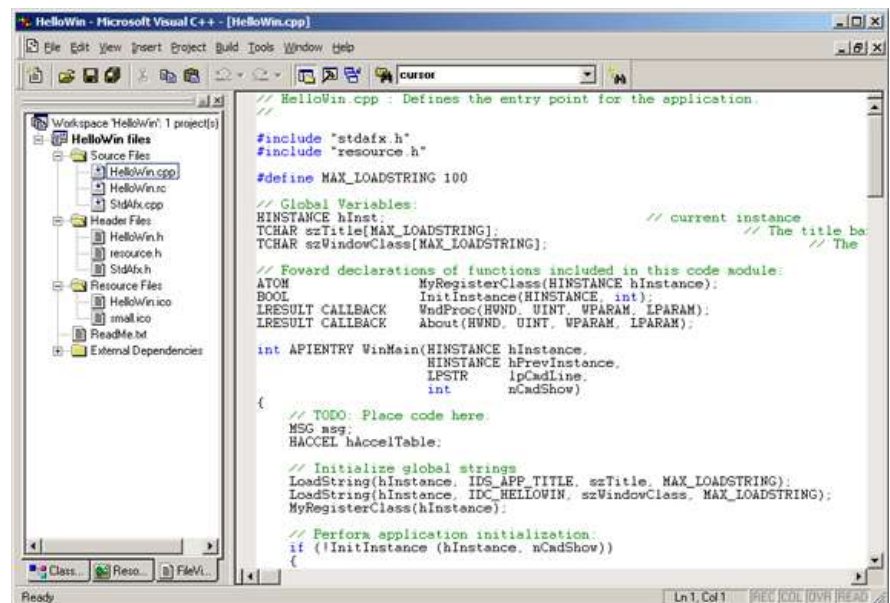
Integrirano razvojno okolje Integrated Development environment (IDE)

- ▶ Na enem mestu zbrana orodja za pisanje, prevajanje in razhroščevanje.
- ▶ Upravljanje s projekti (programi, ki so sestavljeni iz večih datotek).
- ▶ Preprosto razhroščevanje.

WinIDEA je IDE za š-ARM.



Okolje ECLIPSE



Okolje Microsoft Visual Studio

Razhroščevanje na nivoju signalov

Logični analizator

- ▶ Podoben osciloskopu, snema digitalne signale od trenutka proženja.
- ▶ Omogoča iskanje napak v signalih (glitch, motnje, odboji, ..)
- ▶ Ločljivost cca. 20ps (50GHz), nekaj 10 do več kot 100 kanalov
- ▶ Posnetki dolgi več 100MB
- ▶ Razumejo protokole, kot je naprimer PCI Express, ...



Sled programa (Trace)

- ▶ Dnevnik dogajanja v programu
- ▶ Vrednosti izbranih spremenljivk opazujemo v sledilnih točkah (tracepoint)

```
i=2;  
p=1;  
// sledila točka 1  
while (i<=6) {  
    p=p*i;  
    // sledila točka 2  
    i=i+1;  
    // sledila točka 3  
}  
// sledila točka 4
```

sledilna točka	i	p
1	2	1
2	2	2
3	3	2
2	3	6
3	4	6
2	4	24
3	5	24
2	5	120
3	6	120
2	6	720
4	7	720