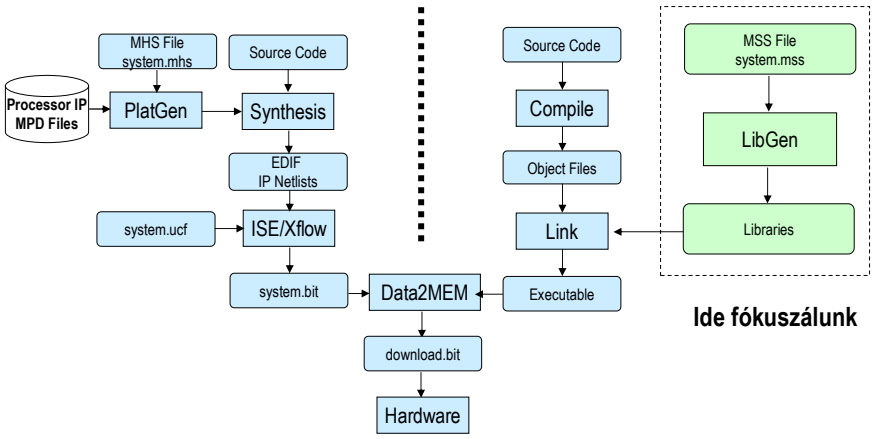
Szoftver tervezés

EDK



Szoftver tervezés
A környezet

- A könyvtár generátor (LibGen) szolgáltatás generálja a beágyazott lágy processzorhoz szükséges könyvtárakat és eszköz meghajtókat
- A LibGen a MSS (Microprocessor Software Specification) fájlt használja bemeneti forrásként. Az MSS fájl határozza meg a perifériákhoz rendelt eszközmeghatókat, a stdin/stdout be/kiviteli perifériát, a megszakítás kezelő rutinokat és más kapcsolódó szoftver jellemzőket
- Az MSS fájlt az XPS rendszer állítja elő a projekt szoftver beállításainak megfelelően

LibGen

A LibGen által generált könyvtárszerkezet

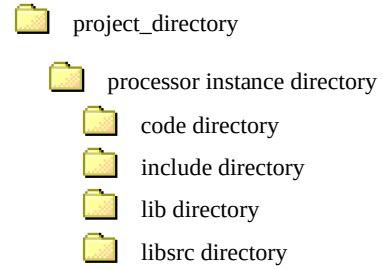
- project_directory
  - Processor instance directory
    - code directory
    - include directory
    - lib directory
    - libsrc directory

Megjegyzés: A processzor példányok szerinti könyvtár annyi példányban jelenik meg, amennyi processzort a rendszerben használunk

- code directory
  - A végrehajtható EDK fájlok tárolója
- include directory
  - C header fájlok, amik a meghajtókhoz szükségesek
  - xparameters.h
    - Definiálja a rendszer perifériáinak bázis és végcímeit
    - Definiálja a periféria ID-ket amiket a meghajtók és a felhasználói programok használnak
    - Definiálja a függvények prototípusait

LibGen

LibGen által generált könyvtárszerkezet



Megjegyzés: A processzor példányok könyvtárai minden LibGen futáskor felülíródnak

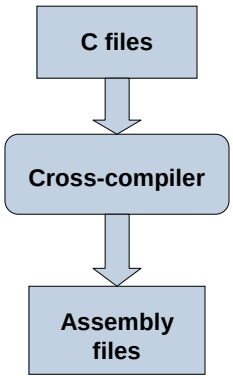
- lib directory
  - **libc.a**, **libm.a** és **libxil.a** könyvtárak
    - A libxil könyvtár tartalmazza azokat a meghajtó függvényeket, amikhez egy konkrét processzor hozzáfér
- libsrc directory
  - Közbenső munka és parancsfájlok, a könyvtárak és meghajtók generálásához
  - Periféria specifikus meghajtó fájlok, amik az EDK és a felhasználói könyvtárakból másolódnak ide

Tartalom

- Bevezetés
- Szoftver beállítások
  - Szoftver Platform beállításai
  - A fordító beállítása
- **GNU eszközök: GCC, AS, LD, Binutils**
- Eszközmeghajtók
  - Level 0, Level 1 szintű támogatás
  - PowerPC Processzor: kivételek, leállítás, időzítés
  - MicroBlaze Processzor: Megszakítások
  - EDK integrált használata
- Könyvtárak
- BSP
  - Indító fájlok és indítási szekvencia

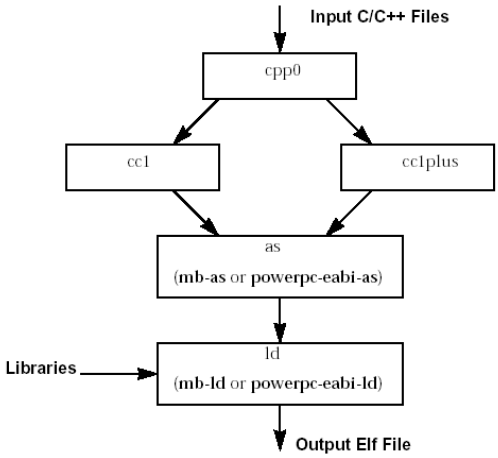
GNU eszközök: GCC

- GCC fordítja a C forráskódot gépi szintű utasításokká
- GCC ezen felül felhasználói interfészt ad a többi GNU eszközhöz (GNU assembler, GNU linker), a megfelelő paraméterekkel indítva ezeket az eszközöket
- A támogatott kereszt-fordítók:
  - PowerPC™ processzor fordító
    - GNU GCC (powerpc-eabi-gcc)
    - Wind River Diab™ fordító (dcc)
  - MicroBlaze™ processzor fordító
    - GNU GCC (mb-gcc)
- Parancssor használat:azokat a beállításokat használja, amiket a GUI-n keresztül beállítottunk



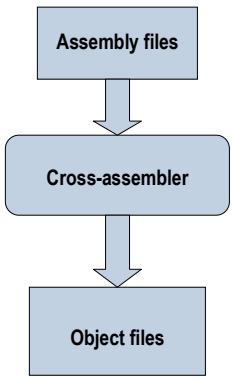
GNU eszközök

- Négy különböző eszközt indít
  - Előfeldolgozó (cpp0)
  - Nyelvspecifikus c-fordító
    - cc1 C program nyelv
    - cc1plus C++ program nyelv
  - Assembler
    - mb-as (MicroBlaze™ processzor)
    - powerpc-eabi-as (PowerPC™ processzor
  - Linker és loader
    - mb-ld (MicroBlaze processzor)
    - powerpc-eabi-ld (PowerPC processzor)



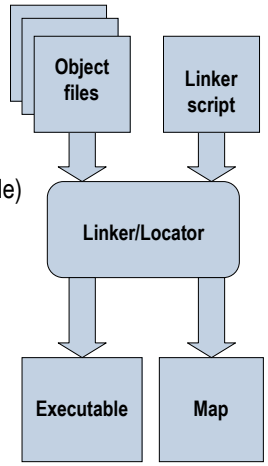
GNU eszközök: AS

- Bemenet: Gépi szintű utasítás fájlok
  - Fájl kiterjesztés: **.s**
- Kimenet: Tárgykód
  - Fájl kiterjesztés: **.o**
  - Tartalma:
    - Összeállított kódrészletek
    - Konstans adatok
    - Külső hivatkozások
    - Fejlesztési információk
- Általában a fordító automatikusan indítja az assemblert
- Használjuk a gcc fordítót a –Wa kapcsolóval ha a c forrást csak lefordítani szeretnénk



GNU eszközök: LD

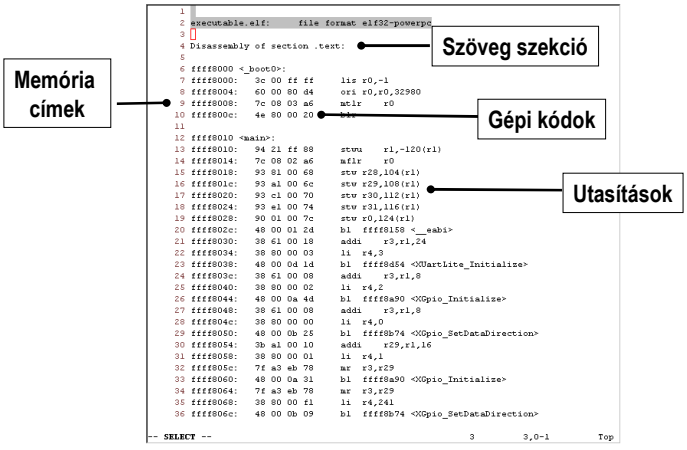
- Linker
- bemenet:
  - Különböző tárgykód fájlok
  - Archivált tárgykód fájlok (könyvtárból)
  - Linker leíró utasítások (elhelyezési fájl, mapfile)
- Kimenet:
  - Végrehajtható memóriafájl (.ELF)
  - Elhelyezési fájl (eltér a bemenetitől)



Binutils: Kódkezelő eszközök

- AR Archiváló
  - Könyvtárak létrehozása, módosítása, elővétel könyvtárból
  - Az EDK ezzel készíti a tárgykód fájlokból az un. Board Support Package (BSP) csomag könyvtárat
  - Az EDK ezzel olvassa ki a tárgykód fájlokat a különböző könyvtárakból
- OBJDUMP
  - Tárgykódok és futtatható kódok különböző információinak megjelenítése
    - Fejléc információ, memória térkép
    - Adatok
    - Gépi kód visszafordítása (Disassemble)
  - GNU futtatható programok
    - powerpc-eabi-objdump
    - mb-objdump

Powerpc-eabi-objdump minta



Eszköz meghajtók

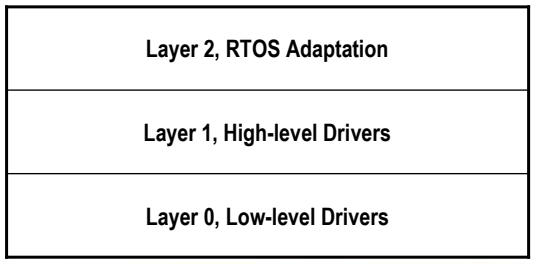
- A Xilinx eszközmeghajtókat a következő célkitűzéseknek megfelelően tervezték:
  - Biztosítsanak maximális hordozhatóságot
    - Az eszközmeghajtók ANSI C forráskódban elérhetők
  - Támogassák az FPGA környezet konfigurálhatóságából származó előnyöket
    - Támogassák a több példányban beépíthető eszközök használatát, a kódrészletek többszörözése nélkül, de biztosítva ugyanakkor az egyes példányok egyedi tulajdonságainak beállíthatóságát
  - Támogassa az egyszerű és összetett alkalmazási eseteket
    - A rétegezett eszközmeghajtó felépítés biztosítja az
      - Egyszerű eszközmeghajtó konfigurációt minimális memória használattal
      - Tejes körű szolgáltatást nagyobb memória méret mellett
  - Könnyű használat és karbantartás
    - Xilinx a szokásos kódolási előírásokat használja és jól dokumentált forráskódokat nyújt a fejlesztők számára

Tartalom

- Bevezetés
- Szoftver beállítások
  - Szoftver Platform beállításai
  - A fordító beállítása
- GNU eszközök: GCC, AS, LD, Binutils
- Eszközmeghajtók
  - **Level 0, Level 1 szintű támogatás**
  - PowerPC Processzor: kivételek, leállítás, időzítés
  - MicroBlaze Processzor: Megszakítások
  - EDK integrált használata
- Könyvtárak
- BSP
  - Indító fájlok és indítási szekvencia

Meghajtók: Alacsony/Magas szint

- A rétegezett felépítés könnyű integrálhatóságot biztosít a következő módon
  - (Layer 2) RTOS alkalmazási réteg
  - (Layer 1) Magas szintű eszközmeghajtó, teljes körű szolgáltatásokkal és hordozhatósággal operációs rendszerek és processzorok között
  - (Layer 0) Alacsony szintű eszközmeghajtó egyszerűbb alkalmazásokra



Eszközmeghajtók: Level 0

- Alacsony szintű eszköztámogatás
- Makrók és függvények, melyek támogatják a kisméretű rendszerek generálását
- Jellemzők:
  - Kis memória igény
  - Minimális vagy semmilyen hibaellenőrzés
  - Csak a legfontosabb eszközfunkciókat támogatják
  - Nincsenek eszköz konfigurációs paraméterek
  - Támogatják az eszközök több példányos használatát a bázis cím kezelésen keresztül
  - Csak lekérdezéses I/O műveletek (nincs megszakítás)
  - Blokkoló függvényhívások
  - A header fájlok nevei tartalmazzák az "_l" kiegészítést (pl. xuartlite_l.h)

Eszközmeghajtók: Level 1

- Magas szintű eszköztámogatás
- Makrók és függvények, melyek támogatják a fejlesztőt az eszközök teljes funkcionalitásának kihasználásában
- Jellemzők:
  - Absztrakt API, ami leválasztja az alkalmazói programozói interfészt a hardver eszköz változásairól
  - Támogatja az eszközök konfigurációs paramétereit
  - Támogatja az eszközök több példányos használatát
  - Lekérdezéses és megszakítást használó I/O műveletek
  - Nem-blokkoló függvényhívások a komplex alkalmazások támogatására
  - Esetleg nagyobb memória hely igény
  - Tipikusan adat puffer interfészeket alkalmaznak az egyszerű bájtos interfészek helyett
  - A header fájlok neve kiegészítés nélküli (pl. xuartlite.h)

Összehasonlítási példa

- **Uartlite Level 1**
  - **XStatus XUartLite_Initialize** (**XUartLite** *InstancePtr, **Xuint16** DeviceId)
  - void **XUartLite_ResetFifos** (**XUartLite** *InstancePtr)
  - unsigned int **XUartLite_Send** (**XUartLite** *InstancePtr, **Xuint8** *DataBufferPtr, unsigned int NumBytes)
  - unsigned int **XUartLite_Recv** (**XUartLite** *InstancePtr, **Xuint8** *DataBufferPtr, unsigned int NumBytes)
  - **Xboolean XUartLite_IsSending** (**XUartLite** *InstancePtr)
  - void **XUartLite_GetStats** (**XUartLite** *InstancePtr, **XUartLite_Stats** *StatsPtr)
  - void **XUartLite_ClearStats** (**XUartLite** *InstancePtr)
  - **XStatus XUartLite_SelfTest** (**XUartLite** *InstancePtr)
  - void **XUartLite_EnableInterrupt** (**XUartLite** *InstancePtr)
  - void **XUartLite_DisableInterrupt** (**XUartLite** *InstancePtr)
  - void **XUartLite_SetRecvHandler** (**XUartLite** *InstancePtr, **XUartLite_Handler** FuncPtr, void *CallBackRef)
  - void **XUartLite_SetSendHandler** (**XUartLite** *InstancePtr, **XUartLite_Handler** FuncPtr, void *CallBackRef)
  - void **XUartLite_InterruptHandler** (**XUartLite** *InstancePtr)
- **Uartlite Level 0**
  - void **XUartLite_SendByte** (**Xuint32** BaseAddress, **Xuint8** Data)
  - **Xuint8 XUartLite_RecvByte** (**Xuint32** BaseAddress)

Mi az a BSP?

- Board Support Package (BSP):
  - A BSP a libxil könyvtárba ágyazott szoftver modul készlet
  - Megengedi a PowerPC™ processzor mag alacsonyszintű funkcióinak használatát
    - A gyorsító tár használat engedélyezése, tiltása, és ürítése
    - Időalap regiszterek írása, olvasása
  - Megengedi az IP periféria eszközök meghajtóinak használatát
    - GPIO, IIC controller, PCI controller, UART
  - Támogatja egyedi kódok és szabvány könyvtárak összefűzését
    - Időzítések, program felfüggesztés
    - Fájlok kezelése
    - Memória kezelés

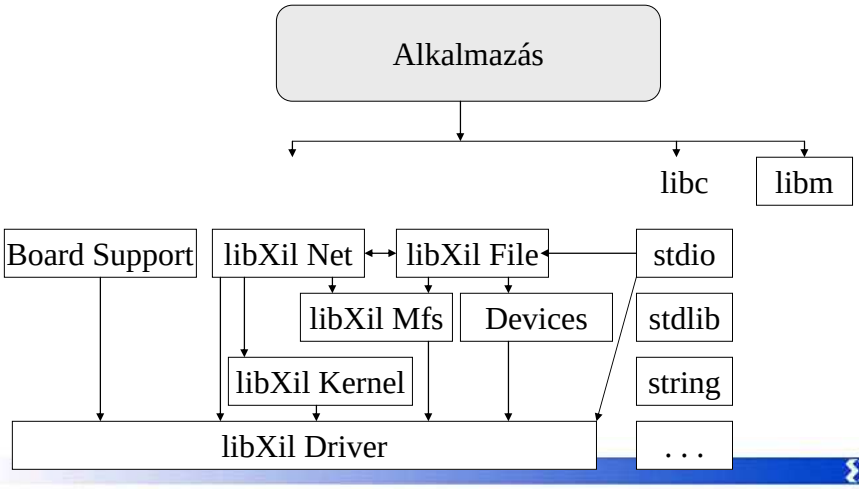
Hardver IP eszközmeghajtók

- Meghajtó
  - Interfészt biztosít a szoftver számára a hardverrel történő kommunikációra
  - Hordozható processzorok és operációs rendszerek között
- Hozzáférhetőség
  - Forráskódban elérhető, ez lehetővé teszi az optimalizálást és egyedi felépítést
  - Minimalizált gépi szintű utasítások
  - C program nyelv

Könyvtárak

- A Xilinx a következő könyvtárakat adja
  - Matematikai könyvtár **(libm)**
    - A math könyvtár a newlib math könyvtár továbbfejlesztése
    - A –lm opciót kell használni a libm függvényekhez
  - Standard C nyelvi támogatás **(libc)**
    - A könyvtár függvényei automatikusan elérhetőek
  - Xilinx C meghajtók és könyvtárak (**libxil)**
    - Xilinx fájlkezelést támogató függvények **LibXil File**
    - Xilinx memória fájl rendszer **LibXil Mfs**
    - Xilinx hálózati támogatás **LibXil Net**

Szolgáltatások



libc

- Standard C függvények: karakterlánc, fájl, időkezelő függvények
- Alap megszakítás és kivétel kezelő függvények
- Egész és lebegőpontos aritmetika
- Dinamikus memóriakezelés: malloc/free (dummy free, a felszabadított memória nem használható újra)
- Speciális input/output:
  - void print(char*)
  - void putnum(int)
  - void xil_printf(const *char fmt, …)

LibXil File

- A LibXil könyvtár blokkos hozzáférést biztosít fájlokhoz és eszközökhöz a **LibXil File** modulon keresztül
- Támogatott függvények:
  - int **open** (const char **name*, int *flags*, int *mode*)
  - int **close** (int *fd*)
  - int **read** (int *fd*, char* *buf*, int *nbytes*)
  - int **write** (int *fd*, char* *buf*, int *nbytes*)
  - int **lseek** (int *fd*, long *offset*, int *whence*)
  - int **chdir** (const char **buf*)
  - const char* **getcwd** (void)

LibXil Mfs

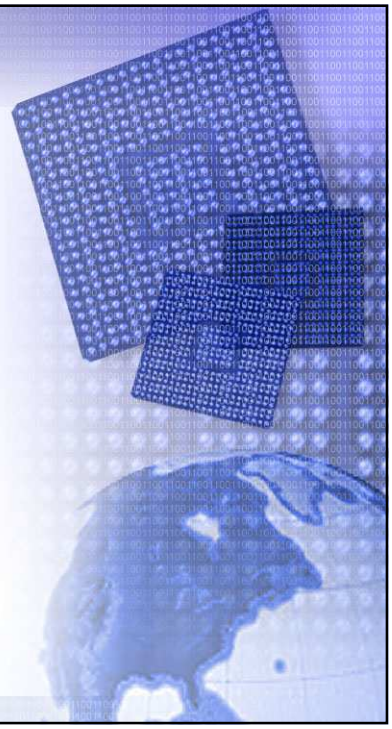
- A memória fájlrendszer (MFS) komponens, **LibXil MFS**, biztosítja a programmemória kezelésének lehetőségét fájlkezelőkön keresztül
  - Könyvtárak nyithatók a memóriában és ezekbe fájlok helyezhetők el
  - A fájlrendszer speciális, fájlrendszer kezelő magas szintű C nyelvű függvényekkel érhető el
- Függvények
  - void **mfs_init_fs** (void)
  - int **mfs_change_dir** (char **newdir*)
  - int **mfs_delete_file** (char **filename*)
  - int **mfs_create_dir** (char **newdir*)
  - int **mfs_delete_dir** (char **newdir*)
  - int **mfs_rename_file** (char **from_file*, char **to_file*)

LibXil Net

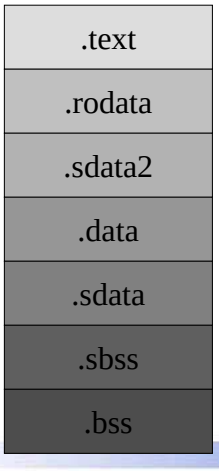
- A hálózatkezelő könyvtár, **libXilNet**, lehetővé teszi, hogy a processzor kapcsolódjon az Internethez. A LibXilNet tartalmazza a TCP/IP protokollt kezelő függvényeket
- Egyszerű programozói interfészt (API) biztosít a hálózati alkalmazásokhoz
- LibXilNet támogatja a többszörös kapcsolódást (a rétegezett interfészen keresztül) a több kliens kiszolgálásához
- Függvények
  - int **xilsock_init** (*void*)
  - int **xilsock_recvfrom** (int *s,* unsigned char* *buf,* int *len*)
  - int **xilsock_sendto** (int *s,* unsigned char* *buf,* int *len*)
  - int **xilsock_recv** (int *s,* unsigned char* *buf,* int *len*)
  - int **xilsock_send** (int *s,* unsigned char* *buf,* int *len*)

XILINX®

Linker script



MicroBlaze Szekciók



Végrehajtható kód, közvetlen értékek

Inicializált konstans adatok, 8 byte-nál nagyobb méretűek, abszolút címzés

Inicializált konstans adatok, 8 byte-nál kisebb méretűek, címzés SDA R/O regiszter segítségével (nincs Imm, egy utasítás)

Inicializált statikus adatok, 8 byte-nál nagyobb méretűek, abszolút címzés

Inicializált statikus adatok, 8 byte-nál kisebb méretűek, címzés SDA R/W regiszter segítségével (nincs Imm, egy utasítás)

Nem inicializált statikus adatok, 8 byte-nál kisebb méretűek, címzés SDA R/W regiszter segítségével (nincs Imm, egy utasítás)

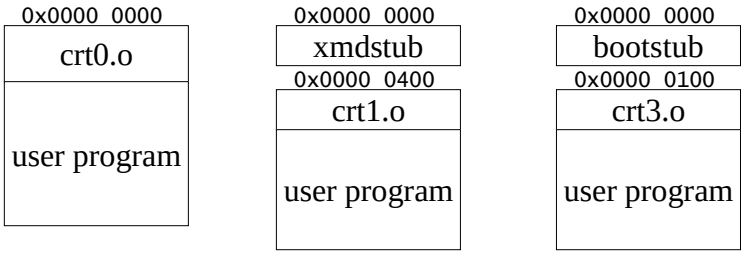
Nem inicializált statikus adatok, 8 byte-nál nagyobb méretűek, abszolút címzés

Minta

```
int ram_data[10] = {0,1,2,3,4,5,6,7,8,9};       /* DATA */

const int rom_data[10] = {9,8,7,6,5,4,3,2,1};   /* RODATA */

int I;   /* BSS */

main(){

...
I = I + 10;  /* TEXT */
...

}
```

Memóriamenedzsment

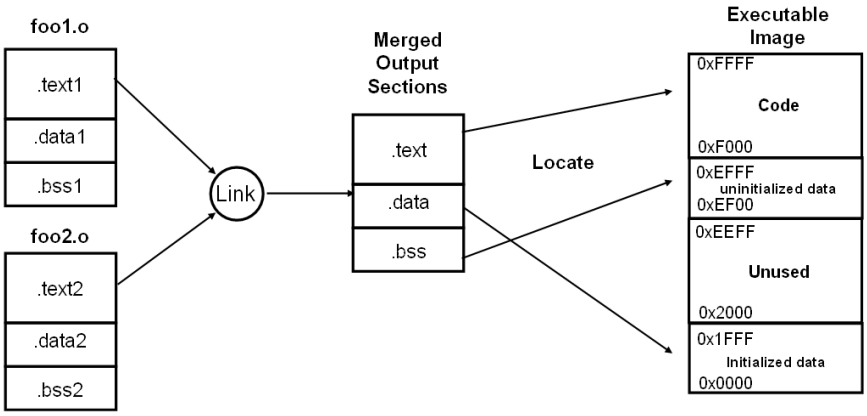
- LMB/OPB memória blokkok, egyéb perifériák (tetszőleges memória kiosztás a 32 bites címtérben, akár hézagokkal is)
- Speciális címek: 0x00 – 0x18 (reset, interrupt, exception)



Object File Sections
Reserved sections that you typically would not modify

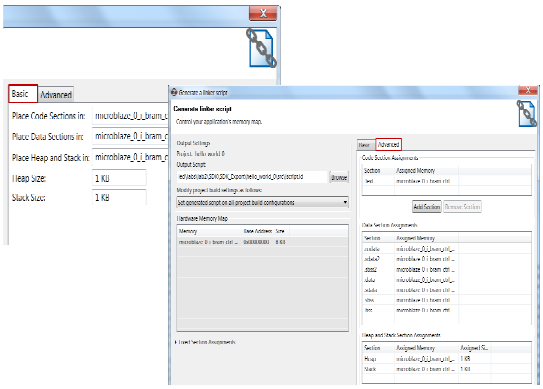
.init	Language initialization code
.fini	Language cleanup code
.ctors	List of functions to be invoked at program startup
.dtors	List of functions to be invoked at program end
.got	Pointers to program data
.got2	Pointers to program data
.eh_frame	Frame unwind information for exception handling

Linker script



Linker Script Generator GUI

- Table-based GUI allows you to define the memory space for code and data sections
- Launch from Xilinx Tools > Generate Linker Script, or from the C/C++ perspective, right-click on <project> > Generate Linker Script
- The tool will create a new linker script (the old script is backed up)



PowerPC Processzor indító fájlok

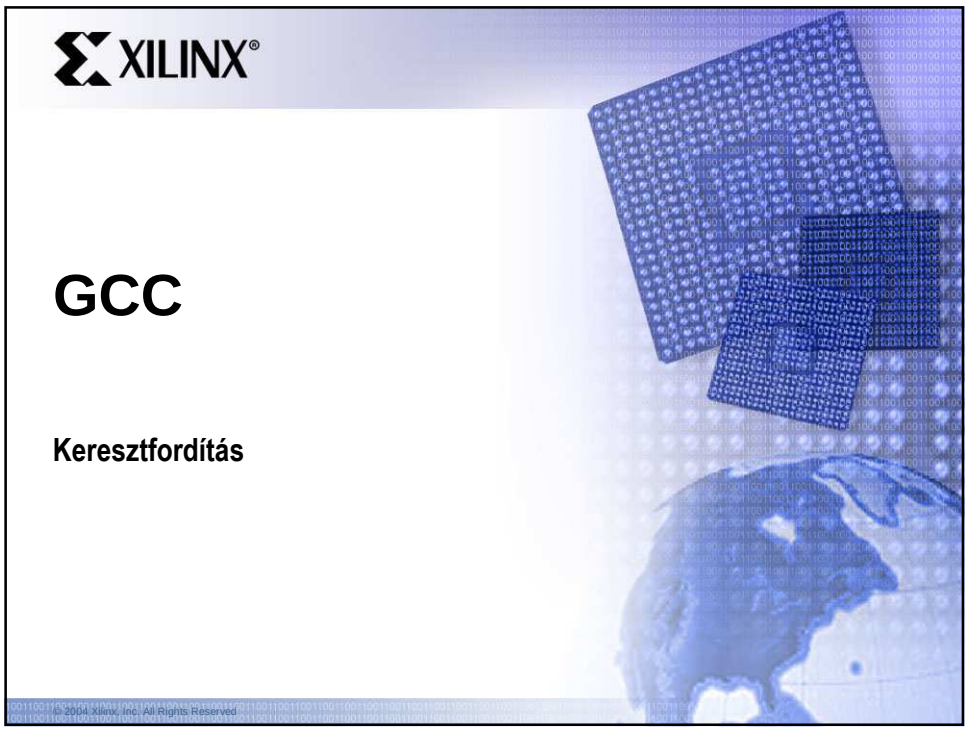
- Fájlok: boot.S, boot0.S, crt0.S, eabi.S
  - Alkalmazások belépési pontja a _**boot** címke a boot.S-ben
  - A _boot pont egy egyszerű ugró utasítás a _**boot0**-ra a boot0.S-ben
  - A _boot0 egy néhány utasításból álló programrészlet, amit egy ugrás zár _**start** címkére a crt0.S-ben
  - _start
    - Törli a .bss és .sbss szekciókat
    - Beállítja a stack-et egy 8 bájtos határra
    - Inicializálja az időalap regisztereket
    - Meghívja a main() függvényt
      - Meghívja az _eabi-t az R13 és R2 regiszterek beállítására, melyek az .sdata és .sdata2 szekciókra mutatnak
      - Végrehajtja a felhasználói feladatokat

PowerPC™

MicroBlaze Processzor Indító fájlok

- Fájl: crt0.S
  - Az alkalmazás belépési pontja a _**start** címke
  - _start
    - Törli a .bss és .sbss szekciókat
    - Beállítja a stack-et egy 8 bájtos határra
    - Inicializálja a kivétel és megszakítás kezelőket
    - Meghívja a main() függvényt

MicroBlaze

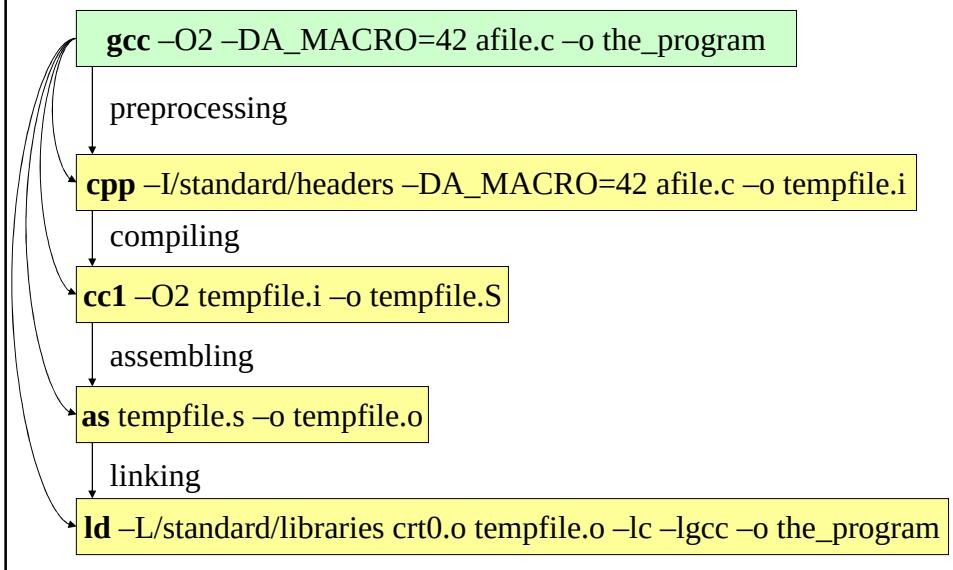


Mi szükséges egy új platform támogatásához

- Keresztfordító - GCC
- Assembler, linker – GNU Binutils
- Debugger – GDB, XMD
- Runtime – libgcc, crt0.S
- C library – libc, libm
- Device drivers - libXil
- Operációs rendszer – XMK, uCOS
- Egyéb eszközök – Data2BRAM, make, stb.

Miért a GNU Compiler ?

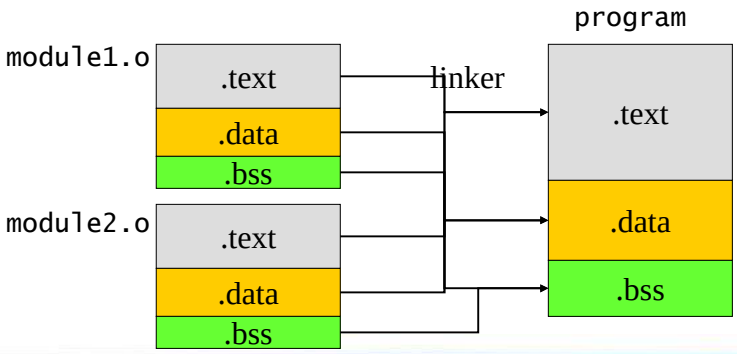
- Fordító készítése bonyolult és időigényes feladat
- Platform és nyelvfüggő megoldások (n x m)
- A GCC rendszer támogat különböző front-end és back-end környezetet
- Nyílt forráskód (a Xilinx hozzájárulása is)



A gcc program csak meghajtja és szinkronizálja a fordítás lépéseit

Szekciók

- .text: programkód
- .data: inicializált statikus adatok
- .bss: inicializálatlan statikus adatok



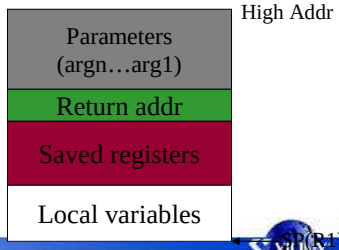
A portolás menete

- ABI specifikáció (Application Binary Interface)
  - C típusok leképzése, memória térkép, függvényhívási protokoll – paraméterek és visszatérési érték átadása, stack kitisztítása
- A platform leírása: néhány C fájl és egy speciális „machine-description” nyelv segítségével

Microblaze ABI

Reg.	Használat
R0	A '0' érték tárolása
R1	Stack Pointer
R2	Konstansok eléréséhez (SDA)
R3-R4	Visszatérési érték
R5-R10	Paraméterek
R11-R12	Átmeneti tároló
R13	Globális változók eléréshez (SDA)
R14	Interrupt visszatérési címe
R15	Szubrutin visszatérési címe
R16	Trap (debugger) visszatérési címe
R17	Kivételkezelő visszatérési címe
R18	Fenntartott az assembler számára
R19-R31	Szabadon használható, de menteni kell tartalmukat
RPC	Programszámláló
RMSR	Státusz regiszer

Típus	Méret (byte)
char	1
short	2
int	4
long	4
enum	4
pointer	2/4



Platformleírók

- **microblaze.h**: C makrók és definíciók az ABI és az operációs környezet leírására
- **microblaze.c**: arhitketúra specifikus függvények megvalósítása
- **microblaze.md**: speciális nyelv (RTL) a processzor leírására

microblaze.h

- A fordító környezete (assembler szintakszis, hol vannak a standard header és library fájlok)
- Alapvető jellegzetességek: big vagy little endian, használható adatméretek, regiszterek, címzési módok
- ABI: hívó vagy a hívott takarít, visszatérési érték regisztere, paraméterek sorrendje, stb.

microblaze.c

- A legtöbb makró megvalósítása a microblaze.h-ból C függvények formájában. Elsősorban a könnyebb debug-olás érdekében

microblaze.md

- Instruction template patterns: tipikus gépi nyelvi elemek leírására és optimalizálásra

Példa: regdst = regsrc1 + regsrc2

```
(define_insn "addsi3"
  [(set (match_operand:SI 0 "register_operand" "=r")
        (plus:SI (match_operand:SI 1 "register_operand" “r")
                 (match_operand:SI 2 "register_operand" “r")))]
  "GET_CODE (operands[2]) != CONST_INT"
  "addk\\t%0,%1,%2"
  [(set_attr "type"    "arith")
   (set_attr "mode"    "SI")
   (set_attr "length"  "1")])
```

microblaze.md

Példa: delay slots

Ha az utasítás típusa “branch, call vagy jump”, felsorolja, hogy mely utasítások

- hajtódnak végre minden esetben
- hajtódnak végre ha a ugrás történik
- hajtódnak végre ha nem történik ugrás

```
(define_delay (eq_attr "type" "branch,call,jump")
  [(eq_attr "type" "!branch,call,jump,icmp,multi,n
o_delay_arith,no_delay_load,no_delay_store,no_delay_xfer,no_del
ay_hilo,no_delay_imul,no_delay_move,darith") (nil) (nil)])
```