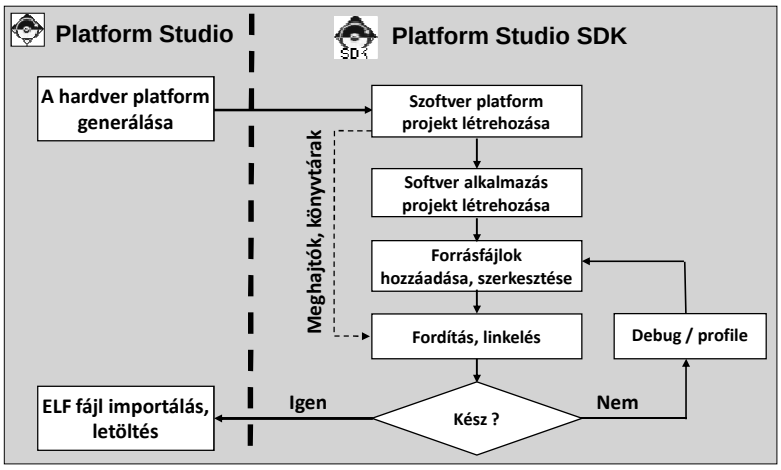
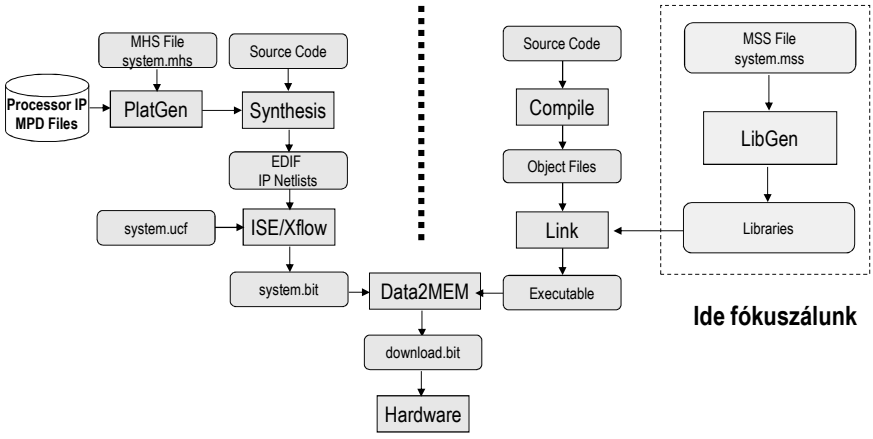
Szoftver tervezés

Software Development Kit (SDK)

A fejlesztés folyamata:



EDK



Ide fókuszálunk

Szoftver tervezés

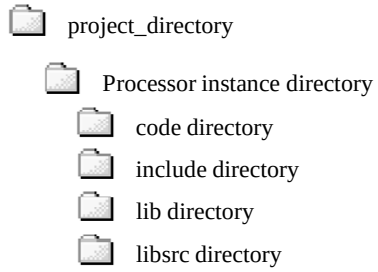
- A könyvtár generátor (LibGen) szolgáltatás generálja a beágyazott lágy processzorhoz szükséges könyvtárakat és eszköz meghajtókat
- A LibGen az **MSS (Microprocessor Software Specification)** fájlt használja bemeneti forrásként. Az MSS fájl határozza meg a perifériákhoz rendelt eszközmeghatókat, a 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

```
BEGIN OS
 PARAMETER OS_NAME = standalone
 PARAMETER OS_VER = 3.06.a
 PARAMETER PROC_INSTANCE = microblaze_0
 PARAMETER STDIN = RS232_Uart_1
 PARAMETER STDOUT = RS232_Uart_1
END
```

```
BEGIN PROCESSOR
 PARAMETER DRIVER_NAME = cpu
 PARAMETER DRIVER_VER = 1.13.a
 PARAMETER HW_INSTANCE = microblaze_0
END
BEGIN DRIVER
 PARAMETER DRIVER_NAME = uartlite
 PARAMETER DRIVER_VER = 2.00.a
 PARAMETER HW_INSTANCE = RS232_Uart_1
END
```

LibGen 1

A LibGen által generált könyvtárszerkezet

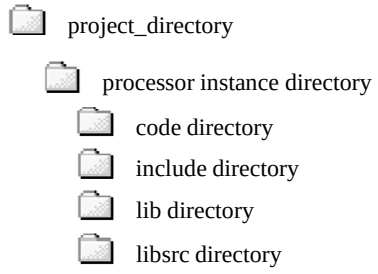


Megjegyzés: A processzor példányok szerinti könyvtár annyi példányban jelenik meg, amennyi processzort a rendszerben használunk

- 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 2

LibGen által generált könyvtárszerkezet

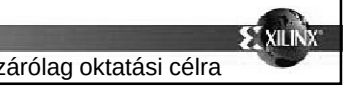


Megjegyzés: A processzor példányok könyvtárai minden LibGen futáskor felülíródnak

- 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

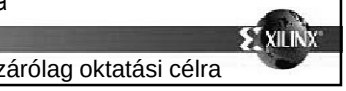
Eszköz meghajtók

- 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

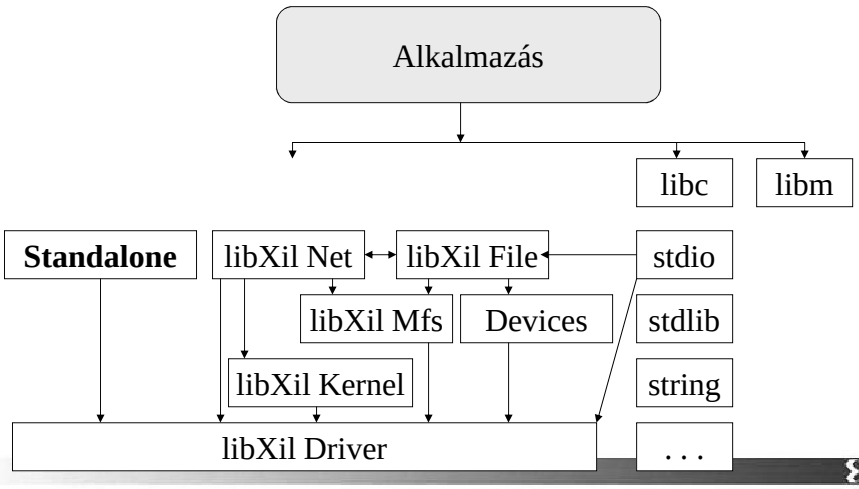
- Bevezetés
- Szoftver beállítások
  - Szoftver Platform beállításai
  - A fordító beállítása
- Eszközmeghajtók
  - **Level 0, Level 1 szintű támogatás**
  - PowerPC Processzor: kivételek, leállítás, időzítés
  - MicroBlaze Processzor: Megszakítások
  - EDK integrált használata
- Könyvtárak
- BSP
  -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



Eszközmeghajtók

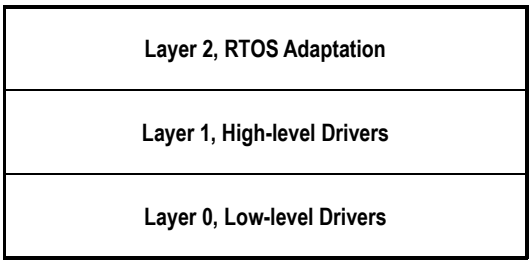
Board Support Package

Szolgáltatások



Alacsony/Magas szintű driver

- 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 színtű 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)

Standalone BSP

- MicroBlaze Processor Interrupt Handling
- MicroBlaze Processor Exception Handling
- MicroBlaze Processor Instruction/Data Cache Handling
- MicroBlaze Processor Fast Simplex Link (FSL) Interface Macros
- MicroBlaze Processor Pseudo-asm Macro Summary
- MicroBlaze Processor Version Register (PVR) Access Routine and Macros

Kivétel források

Exception ID	Value	Description
XEXC_ID_FSL	0	FSL bus exceptions.
XEXC_ID_UNALIGNED_ACCESS	1	Unaligned access exceptions.
XEXC_ID_<BUS>_EXCEPTION(1)	2	Exception due to a timeout from the Instruction side system bus. ***Note:*** *BUS* can be OPB or PLB
XEXC_ID_ILLEGAL_OPCODE	3	Exception due to an attempt to execute an illegal opcode.
XEXC_ID_D<BUS>_EXCEPTION(1)	4	Exception due to a timeout on the Data side system bus. *BUS* can be OPB or PLB
XEXC_ID_DIV_BY_ZERO	5	Divide by zero exceptions from the hardware divide.
XEXC_ID_FPU	6	Exceptions from the floating point unit on the MicroBlaze processor. ***Note:*** This exception is valid only on v4.00.a and later versions of the MicroBlaze processor.
XEXC_ID_MMU	7	Exceptions from the MicroBlaze processor MMU. All possible MMU exceptions are vectored to the same handler. ***Note:*** This exception is valid only on v7.00.a and later versions of the MicroBlaze processor.

MB API

MicroBlaze Processor Instruction/Data Cache Handling

void Xil_ICacheEnable(void)
void Xil_ICacheDisable(void)
void Xil_ICacheInvalidate()
void Xil_ICacheInvalidateRange (unsigned int cache_addr , unsigned int cache_size)
void Xil_DCacheFlush()
void Xil_DCacheFlushRange(unsigned int cache_addr , unsigned int cache_len)
void Xil_DCacheInvalidate()

MicroBlaze Processor Fast Simplex Link (FSL) Interface Macros

getfslx(val,id,flags) putfslx(val,id,flags)
tgetfslx(val,id,flags)
getdfslx(val,id,flags) putdfslx(val,id,flags)
tgetdfslx(val,id,flags) tputdfslx(val,id,flags)
fsl_isinvalid(invalid) fsl_iserror(error)

HAL API

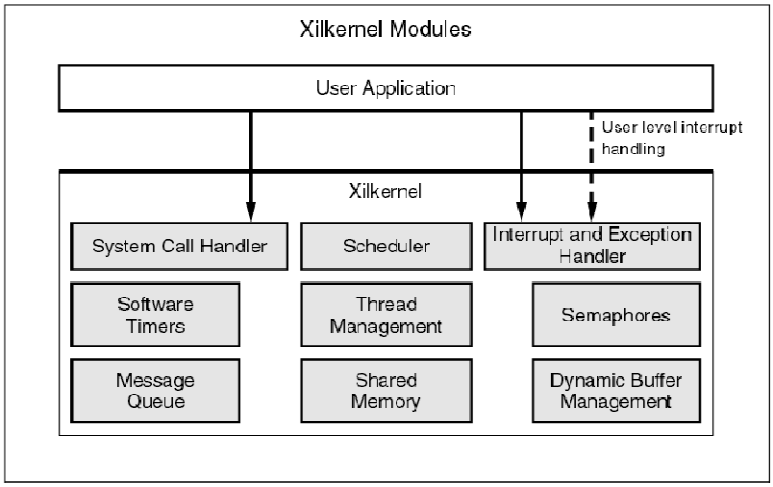
- Types (xil_types)
- Register IO (xil_io)
- Exception (xil_exception)
- Cache (xil_cache)
- Assert (xil_assert)

- Test Memory (xil_testmem)
- Test Register IO (xil_testio)
- Test Cache (xil_testcache)

Xil Kernel BSP

- Complete kernel configuration and deployment within minutes from inside of SDK.
- Robustness of the kernel: system calls protected by parameter validity checks and proper
- return of POSIX error codes.
- Kernel features such as:
- - Threads with round-robin or strict priority scheduling.
- - Synchronization services: semaphores and mutex locks.
- - IPC services: message queues and shared memory.
- - Dynamic buffer pool memory allocation.
- - Software timers.
- - User-level interrupt handling.
- Static thread creation that startup with the kernel
- System call interface to the kernel.
- Exception handling for the MicroBlaze processor.
- Memory protection using MicroBlaze Memory Management (Protection) Unit (MMU)
- features when available

Xil Kernel

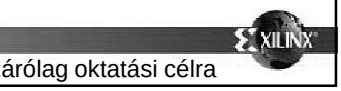


Xilinx könyvtárak

BSP

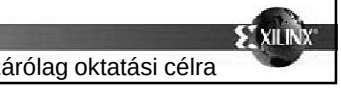
Mi az a BSP?

- 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



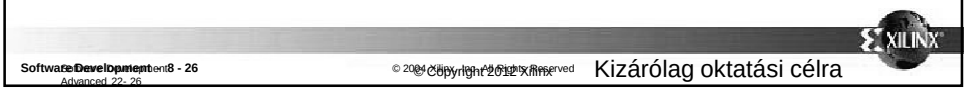
Hardver IP eszközmeghajtók

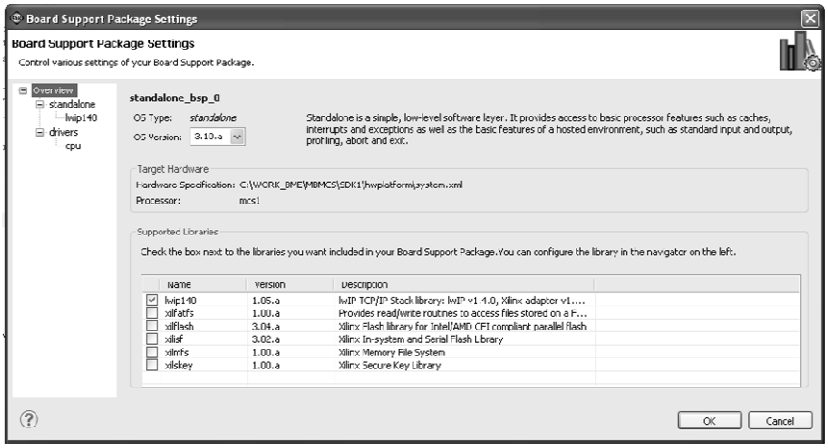
- Meghajtó
  - Interfészt biztosít a szoftver számára a hardverrel történő kommunikációra
  - Hordozható processzorok és operációs rendszerek között
- Hozzáférhetőség
  - Forráskódban elérhető, ez lehetővé teszi az optimalizálást és egyedi felépítést
  - Minimalizált gépi szintű utasítások
  - C program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



Libraries

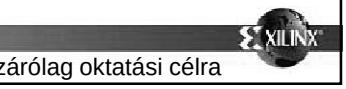
- Xilinx provides three libraries
  - Math library **(libm)**
    - The math library is an improvement over the newlib math library
    - The -lm option is used for libm functions
  - Standard C language support **(libc)**
    - The functions of this library are automatically available
  - Xilinx C drivers and libraries (**libxil)**
    - Xilinx file support functions: **Fatfs**
    - Xilinx memory file system: **Mfs**
    - Xilinx networking support: **lwip**
    - Xilinx flash memory support: **Flash**
    - Xilinx In-system and Serial Flash System: **isf**





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, …)



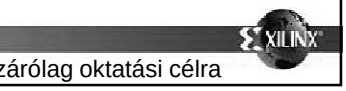
XilFatFS

The XilFATFS filesystem access library provides read/write access to files stored on a Xilinx **System ACE** compact flash or **microdrive**

Hardware platform:

- XPS/AXI System ACE Interface Controller - Logicore module
- System ACE controller and CompactFlash connector
- CompactFlash card or IBM Microdrive formatted with a FAT12, FAT16, or FAT32 file system

```
void *sysace_fopen(const char *file, const char *mode)
int sysace_fread (void *buffer, in t_size, int count, void *file)
int sysace_fwrite(void *buffer, int size, int count, void *file)
int sysace_fclose(void *file)
int sysace_mkdir(const char *path)
int sysace_chdir(const char *path)
int sysace_remove_dir(const char *path)
int sysace_remove_file(c onst char *path)
```



Xilflash

The XilFlash library provides read/write/erase/lock/unlock features to access a **parallel flash** device with **"Common Flash Interface"** (CFI) standard interface.

Intel and AMD CFI compliant flash memory devices.

Hardware platform:

- axi_emc, xps_mch_emc, or similar core for accessing the flash.

```
int XFlash_Initialize (XFlash *I nstancePtr, u32 BaseAddress, u8 BusWidth, int
IsPlatformFlash)
int XFlash_Reset (X Flash *InstancePtr)
int XFlash_Read (XFlash *InstancePtr, u32 Offset, u32 Bytes, void *DestPtr)
int XFlash_Write (XFlash *InstancePtr, u32 Offset, u32Bytes, void *SrcPtr)
int XFlash_Erase (XFlash *InstancePtr, u32 Offset, u32 Bytes)
int XFlash_Lock (XFlash *InstancePtr, u32 Offset, u32 Bytes)
int XFlash_UnLock (XFlash *InstancePtr, u32 Offset, u32 Bytes)
int XFlash_DeviceControl (XFlash *InstancePtr, u32 Command, DeviceControl *Parameters)
int XFlash_IsReady (XFlash *InstancePtr)
```

LibXil Isf

Allows the user to Write, Read, and Erase the **Serial Flash.**

SPI interface drivers in interrupt-driven mode or polled mode for

Supports multiple instances of the same device family

- Atmel, Intel, STM, Winbond, SST, or Spansion

Hardware platform:

- xps_spi, axi_spi, **axi_quad_spi**, ps7_spi, or **ps7_qspi** device

```
int XIsf_Initialize(XIsf *Ins tancePtr, XSpi *SpiInstPtr, u32 SlaveSelect, u8 *WritePtr)
int XIsf_GetStatus(XIsf *InstancePtr, u8 *ReadPtr)
int XIsf_GetStatusReg2(XIsf  *InstancePtr, u8 *ReadPtr)
int XIsf_GetDeviceInfo(XIsf  *InstancePtr, u8 *ReadPtr)
int XIsf_Read(XIsf *Instanc ePtr, XIsf_ReadOperation Oper ation, void *OpParamPtr)
int XIsf_Write(XIsf *InstancePtr, XIsf_WriteOperation Operation, void *OpParamPtr)
int XIsf_Erase(XIsf *InstancePtr, XIsf_E raseOperation Operation, u32 Address)
int XIsf_SectorProtect(XIsf *InstancePtr, XI sf_SpOperation Operation, u8 *BufferPtr)
int XIsf_WriteEnable(XIsf *InstancePtr, u8 WriteEnable)
int XIsf_Ioctl (XIsf *InstancePtr , XIsf_IoctlOperation Operation)
int XIsf_SetSpiConfiguration(XIsf  *InstancePtr, XIsf_Iface *SpiInstPtr, u32 Options, u8 Pres)
inline void XIsf_SetTransferMode(XIsf *InstancePtr, u8 Mode)
```

XilMfs

The LibXil MFS provides the capability to manage program memory in the form of **file handles.**

You can create directories and have files within each directory.

The file system can be accessed from the high-level C language.

```
void mfs_init_fs(int_ numbytes ,_char_ *address ,_int init_type)
void mfs_init_genimage(int numbytes , char  *address, int  init_type)
int mfs_change_dir(char_* newdir)
int mfs_create_dir(char  *newdir)
int mfs_delete_dir(char  *dirname )
int mfs_get_current_dir_name(char  *dirname )
int mfs_delete_file(char *filename )
int mfs_rename_file(char * from_file, char *to_file)
int mfs_exists_file(char * filename )
int mfs_get_usage(int * num_blocks_used, int * num_blocks_free)
int mfs_dir_open(char * dirname)
int mfs_dir_close(int fd)
int mfs_dir_read(int  fd, char_** filename , int *filesize ,int *f iletype)
int mfs_file_open(char * filename , int  mode)
int mfs_file_read(int fd, char  *buf, int  buflen )
int mfs_file_write(int fd, char *buf , int  buflen )
int mfs_file_close(int  fd)
long mfs_file_lseek(int fd, long  offset , int  whence)
```

lwIP 1

Open source TCP/IP protocol suite available under the BSD license.

Operation modes: Standalone/ XilKernel/ other kernels.

Two APIs

• RAW API: Provides access to the core lwIP stack.

• Socket API: Provides a BSD sockets style interface to the stack.

Hardware platform:

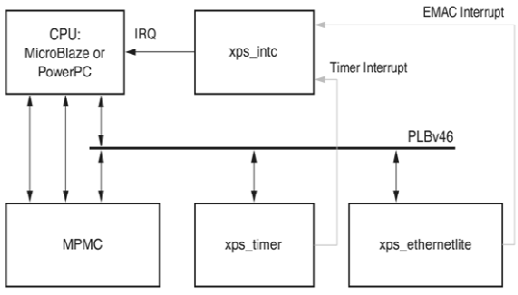
Ethernet lite (xps_ethernetlite, axi_ethernetlite)

- **TEMAC (xps_ll_temac, axi_ethernet)**
- **Zynq (GigE)**

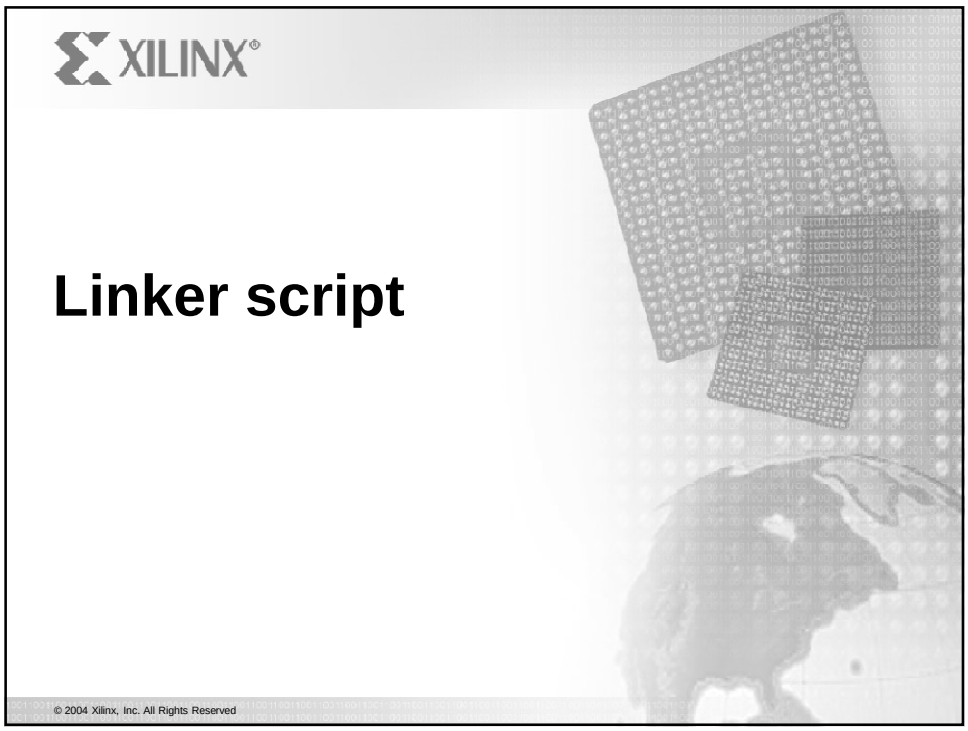
LwIP 2

- **Supported protocols:**
- **• Internet Protocol (IP)**
- **• Internet Control Message Protocol (ICMP)**
- **• User Datagram Protocol (UDP)**
- **• TCP (Transmission Control Protocol (TCP)**
- **• Address Resolution Protocol (ARP)**
- **• Dynamic Host Configuration Protocol (DHCP)**
- **• Internet Group Message Protocol (IGMP)**

LwIP Hardver



FPGA	CPU	EMAC	System Frequency	Max TCP Throughput in RAW Mode	
				Rx Side	Tx Side
Virtex®	MicroBlaze	axi_ethernet	100 MHz	182 Mbps	100 Mbps
Virtex	MicroBlaze	xps_ll_temac	100 MHz	178 Mbps	100 Mbps
Virtex	MicroBlaze	xps_ethernetlite	100 MHz	50 Mbps	38 Mbps



MicroBlaze Szekciók

.text	Végrehajtható kód, közvetlen értékek
.rodata	Inicializált konstans adatok, 8 byte-nál nagyobb méretűek, abszolút címzés
.sdata2	Inicializált konstans adatok, 8 byte-nál kisebb méretűek, címzés SDA R/O regiszter segítségével (nincs Imm, egy utasítás)
.data	Inicializált statikus adatok, 8 byte-nál nagyobb méretűek, abszolút címzés
.sdata	Inicializált statikus adatok, 8 byte-nál kisebb méretűek, címzés SDA R/W regiszter segítségével (nincs Imm, egy utasítás)
.sbss	Nem inicializált statikus adatok, 8 byte-nál kisebb méretűek, címzés SDA R/W regiszter segítségével (nincs Imm, egy utasítás)
.bss	Nem inicializált statikus adatok, 8 byte-nál nagyobb méretűek, abszolút címzés

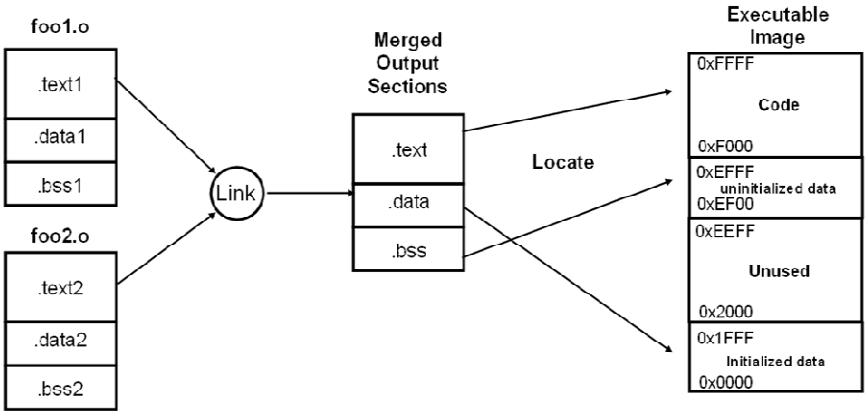
Minta

```
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 */
...

}
```

Object File Sections
Reserved sections that you typically would not modify

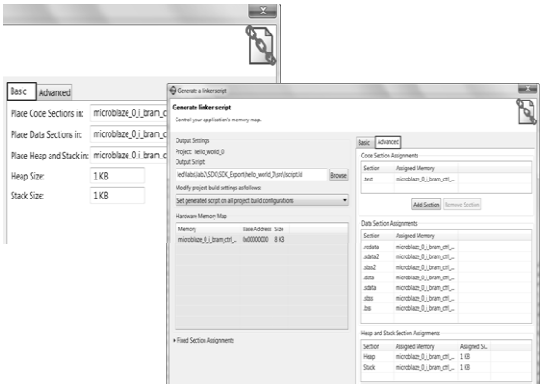
.init	Language initialization code
.fini	Language cleanup code
.ctors	List of functions to be invoked at program startup
.dtors	List of functions to be invoked at program end
.got	Pointers to program data
.got2	Pointers to program data
.eh_frame	Frame unwind information for exception handling

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)



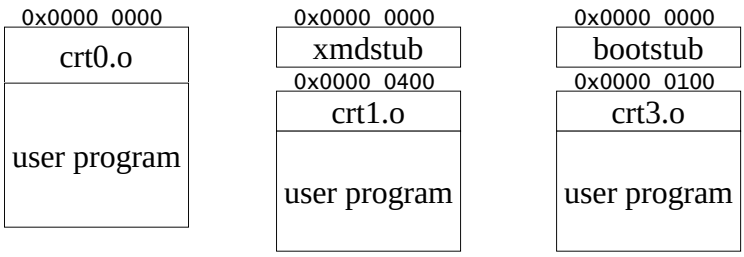
MicroBlaze Processzor Indító fájlok

- 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

Memóriamenedzsment

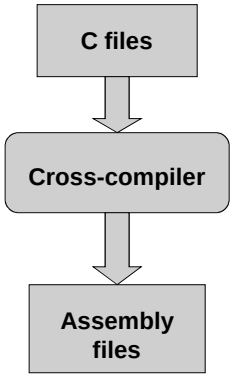
- LMB/OPB memória blokkok, egyéb perifériák (tetszőleges memória kiosztás a 32 bites címtérben, akár hézagokkal is)
- Speciális címek: 0x00 – 0x18 (reset, interrupt, exception)



GNU eszközök

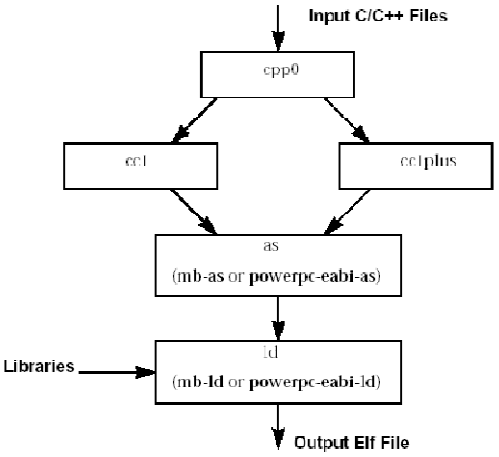
GNU eszközök: GCC

- 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



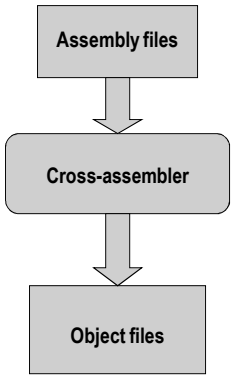
GNU eszközök

- 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)



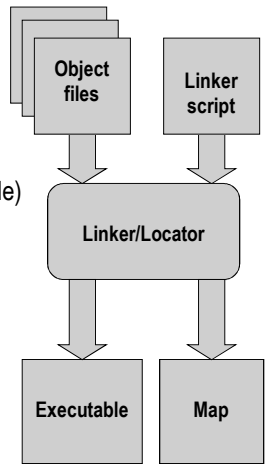
GNU eszközök: AS

- 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

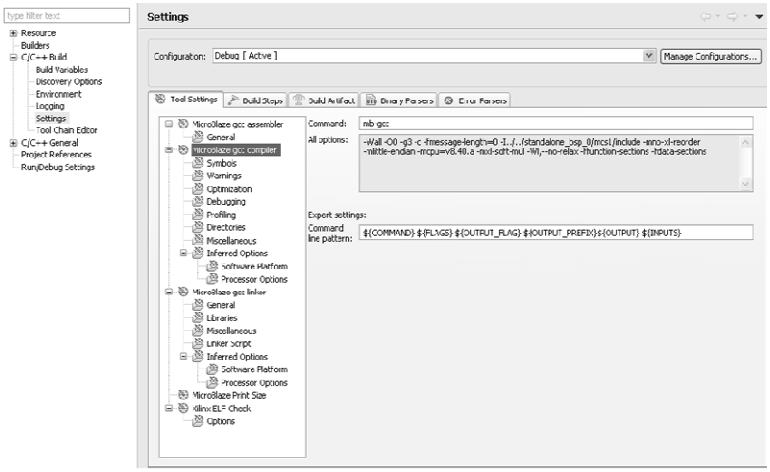


GNU eszközök: LD

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



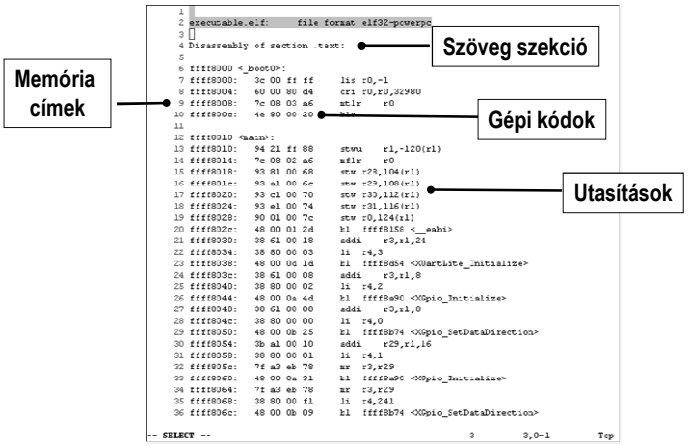
Beállítások az SDK-ban

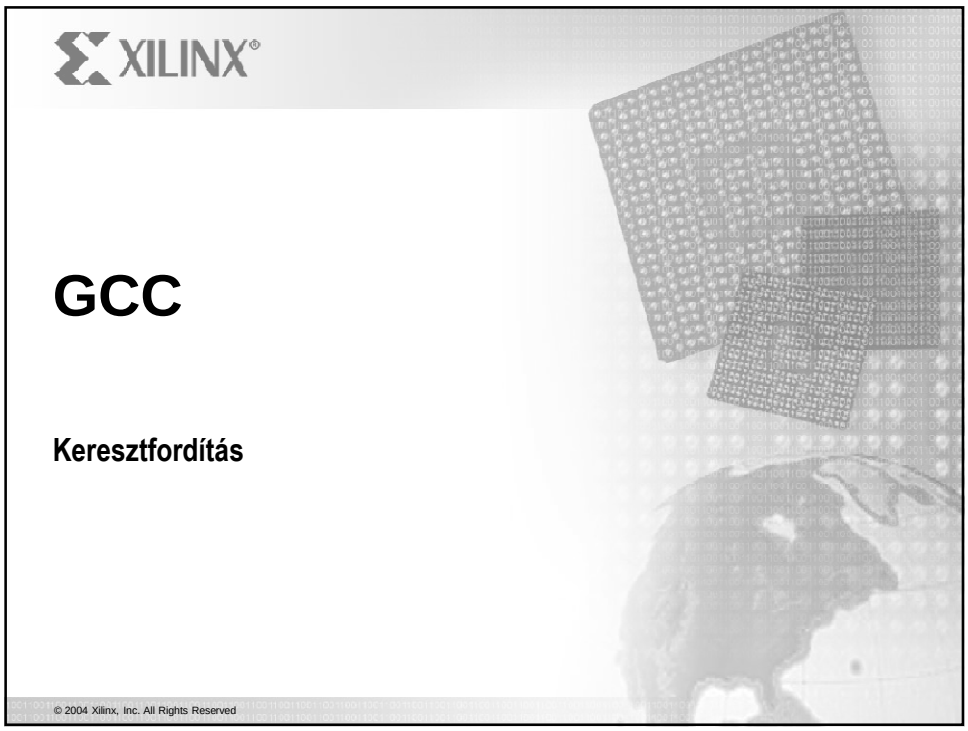


Binutils: Kódkezelő eszközök

- AR Archiváló
  - Könyvtárak létrehozása, módosítása, elővétel könyvtárból
  - Az EDK ezzel készíti a tárgykód fájlokból a 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



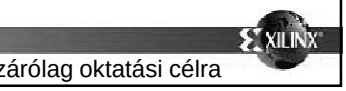


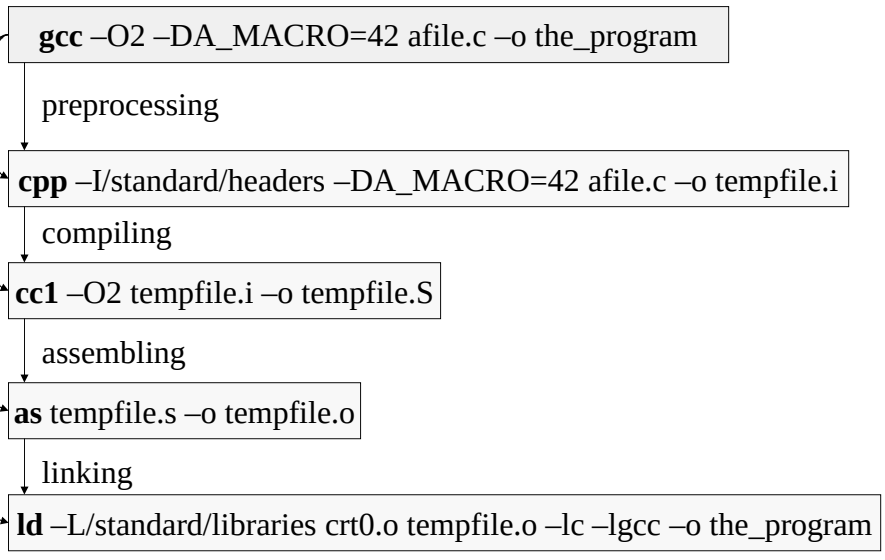
Mi szükséges egy új platform támogatásához

- 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)

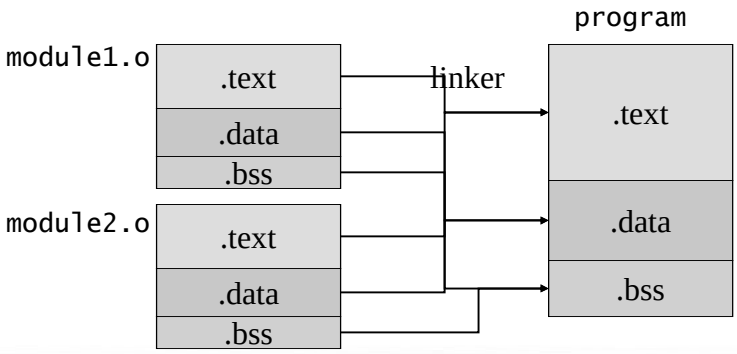




A gcc program csak meghajtja és szinkronizálja a fordítás lépéseit

Szekciók

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



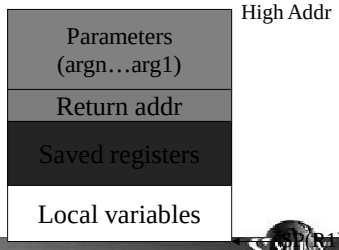
A portolás menete

- 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)])
```