Khomp Platform I
PowerPC 405 GP

LISTA/UFSC
Marcelo Trierveiler Pereira
Prof. Dr. Antônio Augusto Fröhlich
{trier|guto}@lisha.ufsc.br
http://www.lisha.ufsc.br/~{trier|guto}

October 2002
Outline

- The Khomp Platform I
  - Overview
  - Layout
- The PowerPC Processor
  - Overview
  - Registers, I/O Control, Instruction Set
- Compiler
  - PPC Cross-Compiler
- Operating System Functions
  - Context Switch
  - Interrupts
The Khomp Platform I

- **Historic**
  - Developed by Khomp
    - Design house and hardware solution provider
    - Telecommunication embedded system

- **Goals**
  - Integration of telephone and network systems
  - Technologies
    - Telephone: E+M, R2D
    - Network: Ethernet, Wireless, ATM

- **Main design decisions**
  - PowerPC embedded processor (programmability)
  - AD digital signal processor (voice appliances)
Khomp PI Overview

- PPC 405GP
- Ethernet
- PLB
- SDRAM
- PCI
- Flash
- FPGA
- ADSP 2186N
- Voice coprocessor

Frames

Analogic signal
Khomp PI Layout

ENERGY

ETHERNET

PCI

CPLD

FLASH

PPC405GP

SDRAM

SDRAM

FPGA

DSP

DSP Serial BUS

FPGA I/O – E1
Embedded PowerPC

- **Historic**
  - PowerPC was jointly developed by
    - IBM
    - Motorola
    - Apple Computer
  - Based on POWER architecture from RS/6000

- **Main characteristics**
  - Flexible configuration
    - Offers different price and performance
Power PC 405 GP Overview

- System on a chip (SoC) with RISC processor
- Cache
  - 16 KB instruction cache / 8 KB data cache
- Timers
  - Programmable Interval Timer (PIT)
  - Fixed Interval Timer (FIT)
  - Watchdog Timer
- Paging-capable MMU
- 4 KB on-chip RAM
- Device Control Register Bus (DRC)
  - Peripheral control and status
PowerPC 405 GP Block Diagram

Figure 1-1 illustrates the logical organization of the PPC405GP:
PowerPC 405 Data Types

- Data types
  - Bytes (8 bits)
  - Halfwords (16 bits)
  - Words (32 bits)
  - Strings (1 to 128 bytes)
- Instructions are always word-aligned
- Configurable byte ordering
  - Big-endian
  - Little-endian
PowerPC 405 Registers

- 32 general purpose registers-GPRs (0–31)
- Special purpose registers-SPRs
  - 8 x general (spr0–spr7): OS defined
    - Accessed via the mfspr/mtspr instructions
  - Count register (ctr): loop iteration counter
  - Link register (lr): function call return address
  - Fixed point exception register (xer): overflow/carry
  - Processor version register (pvr)
  - Etc...
- 64-bit time stamp counter (tbl/tbu)
  - Accessed via the mftb/mtspr instructions
PowerPC 405 Registers

- Machine status register (msr)
  - Controls interrupts, debugging, etc
  - Accessed via the mfmsr/mtmsr instructions
- 8 x condition registers (cr0−cr7)
  - 4-bits: LT(0), GT(1), EQ(2), SO(3)
- Device control registers (dcr)
  - Accessed via the mfmdcr/mtmdcr instructions
- Interrupt context registers (ssr0−ssr3)
  - Non-critical interrupts: ssr0 ← pc, ssr1 ← msr
  - Critical interrupts: ssr2 ← pc, ssr3 ← msr
PowerPC 405 Instruction Set

- **Storage:** `lwz, stw, etc.`
- **Arithmetic:** `add, subf, neg, mullw (4-cicle), divw (35-cicle), etc`
- **Logical:** `and, or, xor, not, etc`
- **Comparison:** `cmp, etc`
- **Branch:** `b, bc (conditional), bclr (link register), etc`
- **Condition register logical:** `crand, cror, etc`
- **Shift and rotate:** `slw (shift left word), rotlw (rotate left word), etc`
PowerPC 405 Instruction Set

- Cache management: \texttt{dcbi/icbi} (invalidate), \texttt{dcbf} (flush), \texttt{icread/dcread} (read)

- Interruptions
  - Enable/disable: \texttt{mtmsr, mfmsr}
  - Return: \texttt{rfci, rfi}

- Process management
  - System call: \texttt{sc}

- Example: jump
  \begin{verbatim}
  b[l][a] <addr>
  b = branch: PC = PC + <addr>
  l = link register: LR = PC
  a = absolute: PC = <addr>
  <addr> = signed address: 26 bits (+/- 32 MB)
  \end{verbatim}
PowerPC 405 Memory Management

- Memory Management Unit (MMU)
  - Implements paging
    - Variable page size (1K - 16M)
    - Per-page protection and storage attributes
  - Translation Lookaside Buffer (TLB)
    - 64 entries (40 bits log -> 32 bits phy)
    - Logical addresses are associated with \( \text{pid}[24:31] \) register
    - TLB miss triggers interrupts (software management)
  - Controlled via \( \text{msr}[ir,dr] \)
    - Enable/disable for data/instructions
    - Implicitly disabled during interrupts
PowerPC 405 Memory Management

Note: $n$ is determined by page size. See Table 6-1 on page 6-4.
PowerPC 405 I/O

- Device control registers ($dcr$)
  - Defined outside the processor core
  - Control on-chip peripherals and busses
  - Can define an I/O address space or directly map device registers
  - Accessed via the $mf\text{dcr}/m\text{t}\text{dcr}$ instructions

- Memory mapped I/O ($mm\text{i}o$)
  - I/O address space interwoven in the global address space
PowerPC 405 Interrupt Controller

- **Universal interrupt controller (UIC)**

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
<th>DCR</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>UIC0_SR</td>
<td>Status Register</td>
<td>0xC0</td>
<td>when interrupt occurs</td>
</tr>
<tr>
<td>UIC0_ER</td>
<td>Enable Register</td>
<td>0xC2</td>
<td></td>
</tr>
<tr>
<td>UIC0_CR</td>
<td>Critical Register</td>
<td>0xC3</td>
<td></td>
</tr>
<tr>
<td>UIC0_PR</td>
<td>Polarity Register</td>
<td>0xC4</td>
<td></td>
</tr>
<tr>
<td>UIC0_TR</td>
<td>Trigger Register</td>
<td>0xC5</td>
<td></td>
</tr>
<tr>
<td>UIC0_MSR</td>
<td>Masked Status Register</td>
<td>0xC6</td>
<td>UIC0_SR ^ UIC0_ER</td>
</tr>
<tr>
<td>UIC0_VR</td>
<td>Vector Register</td>
<td>0xC7</td>
<td></td>
</tr>
<tr>
<td>UIC0_VCR</td>
<td>Vector Configuration Register</td>
<td>0xC8</td>
<td>critical interrupt only</td>
</tr>
</tbody>
</table>

- **Exception vector prefix register (EVPR):** base
PowerPC 405 Interrupt Handling

- 32 configurable interrupts
  - Non-critical interrupts
    - Handled via special vector registers (offset at 0x0500)
      srr0 <- pc
      srr1 <- msr
      msr <- msr & ~int
      pc <- 0x0500 + EVPR[16:32]
  - Critical interrupts
    - Handled via special vector registers (offset at 0x0100)
      srr2 <- pc
      srr3 <- msr
      msr <- msr & ~int
      pc <- 0x0100 + EVPR[16:32]
      UIC0_VR <- (512*int_no) + UIC0_VCR
PowerPC 405 Initialization

- **After reset**
  - **PC contains** $0xffffffffc$
    - Room for a branch into the actual initialization firmware
  - **MMU, IC, on-chip devices are disabled**
  - **MSR and DCR contains** $0x00000000$
    - Except EBC0_SLOT0 (boot configuration)

- **Initialization firmware**
  - Stored in non-volatile memory
    - Attached to EBC[0] or PCI
  - Configures a 'basic' environment
    - On-chip memory, Interrupt controller, External bus, etc
  - Copy OS code to RAM and jump
PowerPC 405
Operating System Integration

- Initialization
- Interrupt handling
- I/O handling
- Memory management
- Context switch
Initialization

- Succeeds the initialization procedure in firmware
- Main tasks
  - Activate the MMU
  - Activate the Cache
  - Activate timers
  - Scan buses for devices
  - Reconfigure interrupts
  - Loads the OS
  - Creates the first application process
Interrupt Handling

- Critical interrupt example:

```plaintext
srr2 <- pc;
srr3 <- msr;
msr <- msr & ~int;
UIC0_VR <- UIC0_VCR + (512*int_no);
pc <- 0x0100 + EVPR[16:32];

0x100:
mfdcr r0, 0x0c7; //Vector Register
b r0;
```
I/O Handling (MMIO)

- Example: UART
  - Transmitter Holding Register is MMIO at 0xef600300

- Sending a byte through the serial port

```c
char * UART_THR = (char *)0xef600300;
UART_THR = 0xaa;
```
I/O Handling (DCR)

- Example: External Bus Controller (EBC)
  - EBC is formed by 8 slots
  - Configured by 2 DCRs:
    - EBC Config Address: EBC0_CFGADDR = 0x12
    - EBC Config Data: EBC0_CFGDATA = 0x13

```c
char DCR_Addr_Offset = 0x12;
char DCR_Data_Offset = 0x13;
char EBC_Config_Offset = 0x00; // Slot 0
unsigned int Data = 0xffffffff;

mtdcr DCR_Addr_Offset, EBC_Config_Offset;
mtdcr DCR_Data_Offset, Data;
```
Memory Management

- **Process ID Register** \( (\text{pid}) \)
  - Enables work-set preservation across processes
  - 8 bits appended to TLB entries (log. addr.)

- **On context switch**
  - Invalidate TLB or
  - Change \( \text{pid} \)

- **OS must handle**
  - TLB misses
    - Page-table lookup, victim selection, replace
  - Page-faults
    - Demand paging (virtual memory)
    - OS-specific treats (copy-on-write, stack grow, etc)
void context_switch(int **old_sp, int *new_sp) {
    // get old_sp
    asm("lwz r1, 0(r3) \n"

    // save general registers
    "stwu r0,-4(r1) \n"
    :
    "stwu r31,-4(r1) \n"

    // save special registers
    "mfldr r11 \n"
    "stwu r11,-4(r1) \n"
    "mfmsr r11 \n"
    "stwu r11,-4(r1) \n"
    :
    "mfxer r11 \n"
    "stwu r11, -4(r1) \n"

    // sync memory and pipeline
    "sync \n"

    // update old_sp
    "stw r1,0(r3) \n"

    // get new_sp
    "mr r1,r4 \n"

    // load special registers
    "lwzu r11,4(r1) \n"
    "mtxer r11 \n"
    :
    "lwzu r11,4(r1) \n"
    "mtmsr r11 \n"
    "lwzu r11,4(r1) \n"
    "mtlr r11 \n"

    // load general registers
    "lwzu r31,4(r1) \n"
    :
    "lwzu r0,4(r1) \n"

    // sync memory and pipeline
    "sync \n"
Embedded PowerPC Programming

- Cross-compilation
  - Host
    - design, code, compile, debug, etc...
  - Target
    - run, debug

- GCC
  - Embedded Application Binary Interface (EABI)
    - Conventions for interoperability
    - Defines register's context
      - r1 -> stack pointer, r3-r10 -> parameters, r3-r4 -> return values, etc
  - ELF -> SREC
    - SREC is a “raw” memory dump
## PowerPC 405 Registers

### Access Control

#### Supervisor Model

<table>
<thead>
<tr>
<th>User Model</th>
<th>Supervisor Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPR[ 0 – 31 ]</td>
<td>Storage Regs.</td>
</tr>
<tr>
<td>SPR 0</td>
<td>Machine State Reg.</td>
</tr>
<tr>
<td>CR – Cond. Reg</td>
<td>SPRs[ 0 – 7 ]</td>
</tr>
<tr>
<td>XER – Fixed Point Reg.</td>
<td>Exception Regs.</td>
</tr>
<tr>
<td>LR – Link Reg.</td>
<td>Save/Restore Regs.</td>
</tr>
<tr>
<td></td>
<td>Process Id. Reg.</td>
</tr>
<tr>
<td></td>
<td>Debug Regs.</td>
</tr>
<tr>
<td></td>
<td>Timer Regs.</td>
</tr>
<tr>
<td></td>
<td>External Core Regs.</td>
</tr>
</tbody>
</table>
## PowerPC 405

### Device Control Registers

<table>
<thead>
<tr>
<th>Device Control Registers (DCRs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDRAM</td>
</tr>
<tr>
<td>EBC</td>
</tr>
<tr>
<td>DCP</td>
</tr>
<tr>
<td>OCM</td>
</tr>
<tr>
<td>PLB</td>
</tr>
<tr>
<td>OPB</td>
</tr>
<tr>
<td>CPC</td>
</tr>
<tr>
<td>UIC</td>
</tr>
<tr>
<td>DMA</td>
</tr>
<tr>
<td>MAL</td>
</tr>
<tr>
<td>PCI</td>
</tr>
<tr>
<td>UART</td>
</tr>
<tr>
<td>IIC</td>
</tr>
<tr>
<td>GPIO</td>
</tr>
<tr>
<td>EMAC</td>
</tr>
</tbody>
</table>
# PowerPC 405 Default Address Space

<table>
<thead>
<tr>
<th>Function</th>
<th>Start Addr.</th>
<th>End Addr.</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Memory/Peripherals</td>
<td>0x0000 0000</td>
<td>0x7FFF FFFF</td>
<td>2GB</td>
</tr>
<tr>
<td>PCI Space</td>
<td>0x8000 0000</td>
<td>0xEF5F FFFF</td>
<td>1.744GB</td>
</tr>
<tr>
<td>Internal Peripherals</td>
<td>0xEF60 0000</td>
<td>0xEFFF FFFF</td>
<td>10MB</td>
</tr>
<tr>
<td>Expansion ROM</td>
<td>0xF000 0000</td>
<td>0xFFDF FFFF</td>
<td>454MB</td>
</tr>
<tr>
<td>Boot ROM</td>
<td>0xFFE0 0000</td>
<td>0xFFFF FFFF</td>
<td>2MB</td>
</tr>
</tbody>
</table>