- Open in browser: http://172.18.5.168:5000

Agenda

- ARM7 modes
- ARM7 exceptions
- ARM7 registers
- PSR register
- Cortex-A PSR register
- Makefile
- Bitwise operators

ARM7 Programmer's Model

Modes

- Privileged modes
    - Supervisor: Default mode on reset or due to software interrupt.
    - IRQ: Interrupt request.
    - FIQ: Fast Interrupt request (higher priority)
    - Undef: While decoding invalid instruction.
    - Abort: Memory access violation (fetcing instruction at wrong address or read/write data on wrong address).
    - System: Mode for running system software/OS.
- Non-Privileged mode
    - User: Mode for running user program.
- Privileged mode
    - All (relevant) registers are accessible.
    - All instructions are allowed.
    - Exception handling is possible.
    - Usually system software/OS runs in these modes.
- Non-Privileged mode
    - Few registers are not accessible (e.g. SPSR).
    - Few instructions are not allowed (e.g. MSR, MRS).
    - Usually user program runs in this mode.
- The mode is represented by 5 bits [4:0] of CPSR register.
    - e.g. User mode: 10000

Exceptions

- Exception cause program execution to differ from its normal/expected flow.
- In ARM, when exception occurs, current execution is paused. Exception handler (set of

instructions) is executed and then current execution resumes.

- ARM exceptions
    - Reset
    - Software Interrupt
    - Prefetch Abort
    - Data Abort
    - IRQ
    - FIQ
    - Undef
- When exception occurs, mode is auto switched.
    - Reset --> Supervisor
    - Software Interrupt --> Supervisor
    - Prefetch Abort --> Abort
    - Data Abort --> Abort
    - IRQ --> IRQ
    - FIQ --> FIQ
    - Undef --> Undef

Common registers (in all modes)

```
* r0-r7: low general-purpose registers
* r8-r12: high general-purpose registers
* r13: stack pointer (sp)
* r14: link register (lr)
* r15: program counter (pc)
* cpsr: current program status register
```

stack pointer (sp)

- Points to top of stack.

link register (lr)

- When function is called (BL instruction) or exception is raised, the address of next instruction is copied into LR register.
- This is faster than pushing that address on stack (which is common in most of architectures).
- While returning return address copied from LR into PC.
    - mov pc, lr
    - b lr
- However during nested function call or interrupts the return address in LR may be overwritten. In this case, it programmer's responsibility to copy LR into stack.

program counter (pc)

- It contains address of next instruction to be fetched.
- Due to pipeline (3-stage) it will be always 2 instructions ahead of instruction under execution.

- ARM state
    - Each instruction is of 4 bytes. So address of each instruction is multiple of 4 bytes.
    - PC is most significant 30 bits [31:2]. The bits [1:0] are always considered to be 0.
- Thumb state
    - Each instruction is of 2 bytes. So address of each instruction is multiple of 2 bytes.
    - PC is most significant 31 bits [31:1]. The bit [0] are always considered to be 0.
- The last bit of PC is used to change state of processor.
    - 0 -- T=0 -- ARM state
    - 1 -- T=1 -- Thumb state

cpsr register

- Contains current program/processor status
- Four parts
    - control [7:0]
        - mode [4:0] -- 5 bits
        - state [5] -- T bit (ARM/Thumb state)
        - I (irq) [7]
            - When IRQ interrupt occurs, this bit is set to 1.
            - When this bit is set to 1, IRQ interrupts are masked.
        - F (fiq) [6]
            - When FIQ interrupt occurs, this bit is set to 1.
            - When this bit is set to 1, FIQ interrupts are masked.
    - execution [15:8]
        - Reserved in ARMv4
        - abc bits (in ARM v6) -- from abcde bits of IT instructions
        - E: Endianness bit (in ARM v7)
            - Little Endian
                - Lower byte on lower address.
            - Big Endian -- also called as Network order.
                - Lower byte on higher address.
        - A: disable imprecise data aborts
    - status [23:16] (in ARM v7)
        - Reserved in ARMv4
        - four GE bits -- SIMD instructions (in ARM v7)

    - flagss
        - ALU flags [31:28] -- NZCV
        - Q [27] (in ARM v5) -- sticky bit
            - Saturated math instructions (DSP instructions)
        - J [24] (in ARM v5) -- Jazzelle state
            - When J=1 and T=0, core enters in Jazzelle state.
            - It executes Java byte code.
        - de bits (in ARM v6) -- from abcde bits of IT instructions

FIQ registers

- r8-r12: dedicated registers for FIQ.
- lr,sp: dedicated registers for FIQ.
- spsr: Saved Program Status Register.
    - Contains copy of CPSR when state is changed.

Other modes registers

- IRQ, Supervisor, Abort, Undef, User mode.
- System mode have same registers as of User mode.
- lr, sp: dedicated registers for each mode.
- spsr: dedicated register for each mode.

Check Endianness

```c
short a = 0x1122;
char *p = (char*)&a;
if(*p == 0x22)
    printf("Little endian\n");
else
    printf("Big endian\n");
```

```c
union {
    short a;
    char b[2];
}v;
v.a = 0x1122;
if(b[0] == 0x22)
    printf("Little endian\n");
else
    printf("Big endian\n");
```

Makefile

- To build programs compilation commands are given. For compiling big programs (multi-

file) multiple commands are required.
- To simplify building such huge programs, "make" utility is used.
- The "make" command reads a file containing commands for compilation/linking, that file is called as "makefile".
- The make command only compiles modified source files and files dependent on them (instead of compile all files every-time). It track the changes using time-stamp of the file.
- The name of makefile can be makefile, Makefile, GNUMakefile or any custom name. For custom filename, it should be specified to make command.

    - make -f custom-makefile

- Makefile contains commands for compilation and linking -- called as "rules". Also they contain, which (output) file is dependent on which (input) file -- called as "dependencies".

Makefile Ex1

- hello.c

```c
// hello.c
#include <stdio.h>
int main()
{
    printf("Hello C!\n");
    return 0;
}
```

- on terminal give following commands.

```
gcc -c hello.c

gcc -o hello.out hello.o

./hello.out
```

- Makefile

```
hello.out: hello.o
    gcc -o hello.out hello.o

hello.o: hello.c
    gcc -c hello.c

```

- On terminal

```
./hello.out
```

Makefile Ex2

- add.h

```
int add(int a, int b);
```

- add.c

```
int add(int a, int b)
{
    return a + b;
}
```

- subtract.h

```
int subtract(int a, int b);
```

- subtract.c

```
int subtract(int a, int b)
{
    return a - b;
}
```

- multiply.h

```
int multiply(int a, int b);
```

- multiply.c

```
int multiply(int a, int b)
{
    return a * b;
}
```

- main.c

```
#include <stdio.h>
#include "add.h"
#include "subtract.h"
#include "multiply.h"

int main()
{
    int res;
    res = add(23, 5);
    printf("add result : %d\n", res);
    res = subtract(23, 5);
    printf("subtract result : %d\n", res);
    res = multiply(23, 5);
    printf("multiply result : %d\n", res);
    return 0;
}
```

- Makefile

```
main.out: add.o subtract.o multiply.o main.o
    gcc -o main.out add.o subtract.o multiply.o main.o

add.o: add.c
    gcc -c add.c

subtract.o: subtract.c
    gcc -c subtract.c

multiply.o: multiply.c
    gcc -c multiply.c

main.o: main.c
    gcc -c main.c
```

gcc

- gcc -- GNU C Compiler
    - It is part of GNU Compiler Collection.
    - Front-end for compilation tools i.e. preprocessor, compiler, assembler and linker.
    - gcc internally calls cpp, cc1, as and ld in sequence.

Compilation

- -c: Compile only (do not link)
- -E: Preprocess (produce expanded source code .i)
- -S: Produce assembly code (.S)
- -I: Path of standard include directories. -I/path/of/include-dir
- -D: define macros & symbols
    - e.g. -DPI=3.142
    - e.g. -D_GNU_SOURCE
- -o: Name of output file.
- -O: Optimization option. -Os
    - 0 (no optimization), 1, 2, 3 (max optimization), s (optimization for size)
- -g: Enable debugging and specify debug level/symbols.
    - x86
        - gdb1 (minimal info)
        - gdb2
        - gdb3 (maximum info)
    - embedded
        - warf
        - stabs

- -m: specify target machine
    - e.g. -m32 : 32 bit compilation

Linking

- -o: Name of output file.
- -L: standard library dir path
- -lxyz: link with library libxyz.so
    - e.g. -lc -- link with libc.so
    - e.g. -lm -- link with libm.so
    - e.g. -lpthread -- link with libpthread.so

```
# create .i file from .c file
gcc -E -o hello.i hello.c

# create .s file from .c file
gcc -S hello.c

# create .o file
gcc -c hello.c

# create .out file from .o
gcc -o hello.out hello.o

# create .out file from .c
gcc -o hello.out hello.c

# create .out file from .c -- also add debug info
gcc -ggdb1 -o hello.out hello.c
gcc -ggdb2 -o hello.out hello.c
gcc -ggdb3 -o hello.out hello.c

# create .out file from .c -- also optimize for size
gcc -Os -o hello.out hello.c

# create .out file from .c -- also define macro for PI
gcc -DPI=3.14 -o hello.out hello.c
```