Lecture 8

Processor Modes & Thumb Code

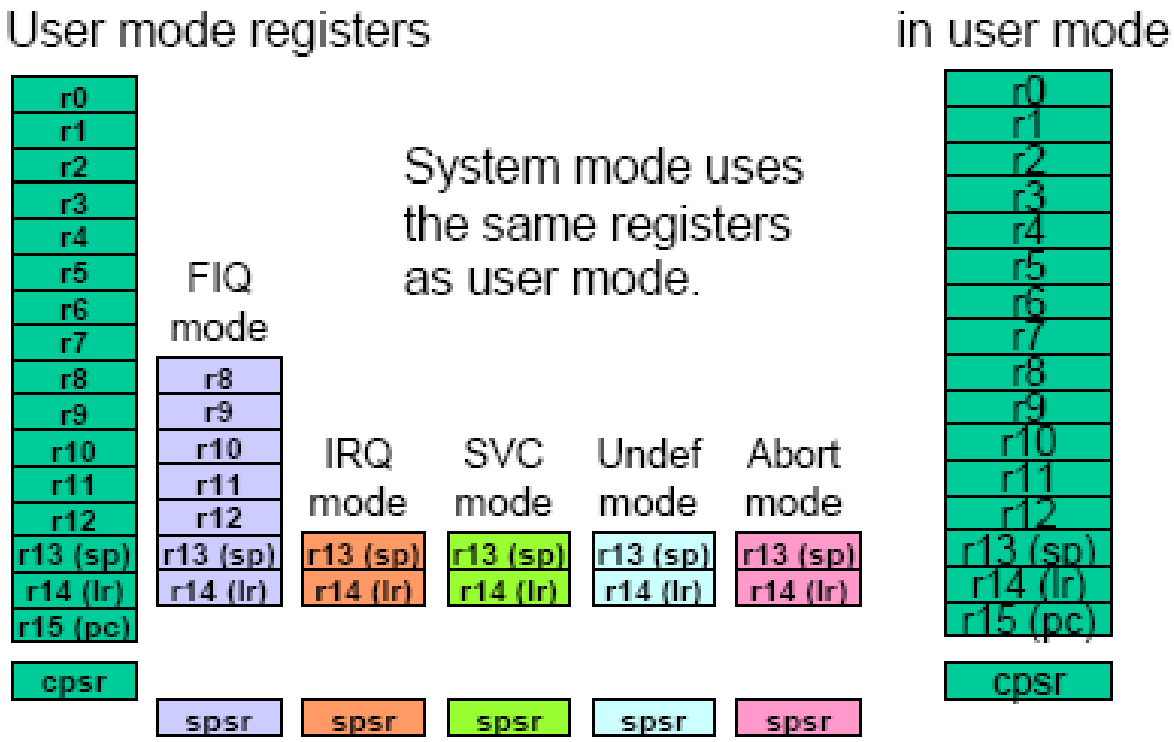
ARM7 – processor modes

- The ARM processor has seven ‘processor modes’ – we have already covered three of these;
  - User mode,
  - FIQ mode and
  - IRQ mode.
- The other modes are
  - Supervisor,
  - Abort,
  - Undef and
  - System.
- Supervisor mode is entered on reset or when a software interrupt (SWI) instruction is executed. This mode is used for operating system type functions and SWI is also known as a ‘supervisor call’.

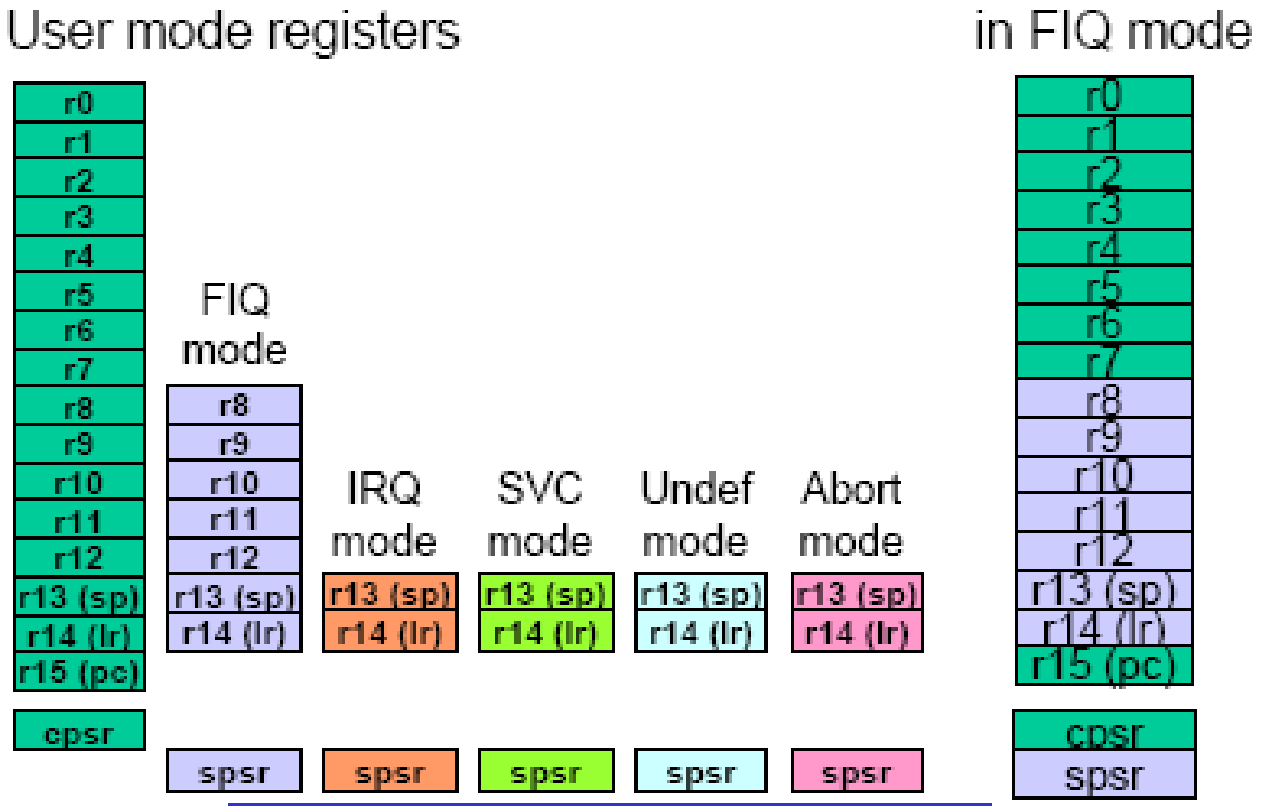
ARM7 – processor modes

- Abort mode is used when a memory access violation occurs
  - either on an instruction fetch
  - or on a data read/write
  - e.g. when a part of memory is not immediately available because it has stored on hard disk.
- Undef mode is entered when the instruction decoder encounters machine code for which there is no instruction
  - the machine code is undefined.
- System mode is a privileged mode used for the operating system.

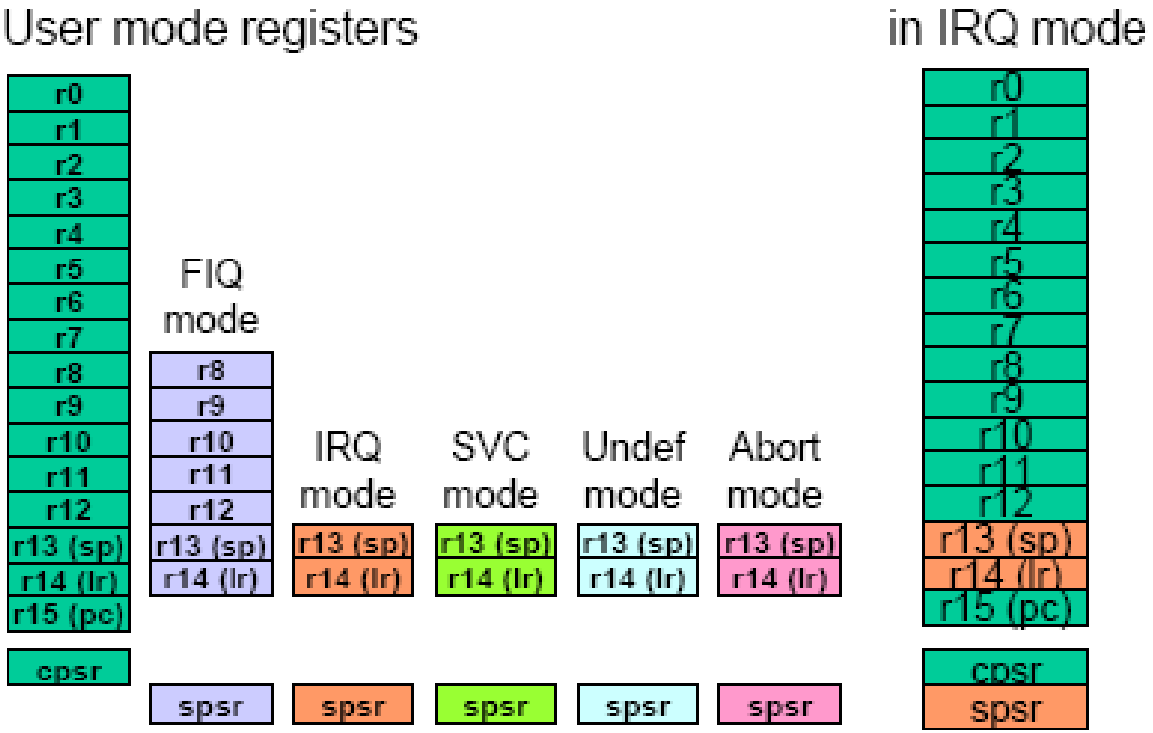
Different registers in each mode



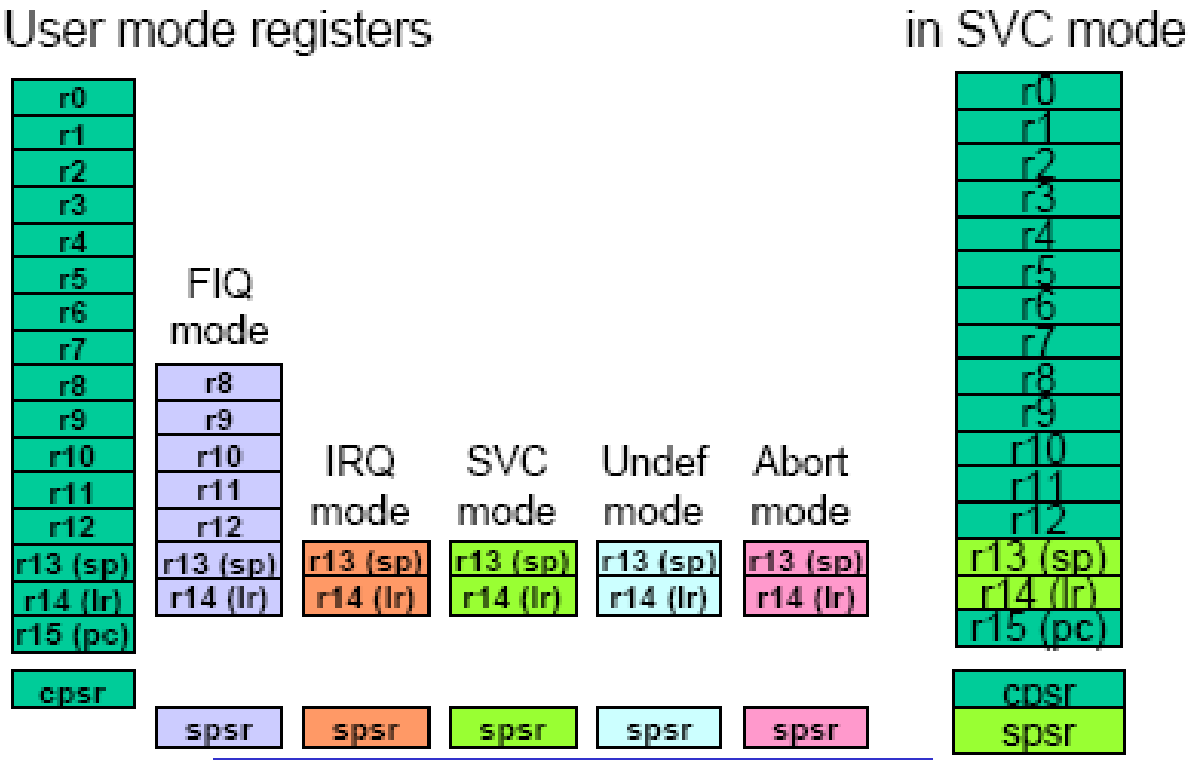
Different registers in each mode



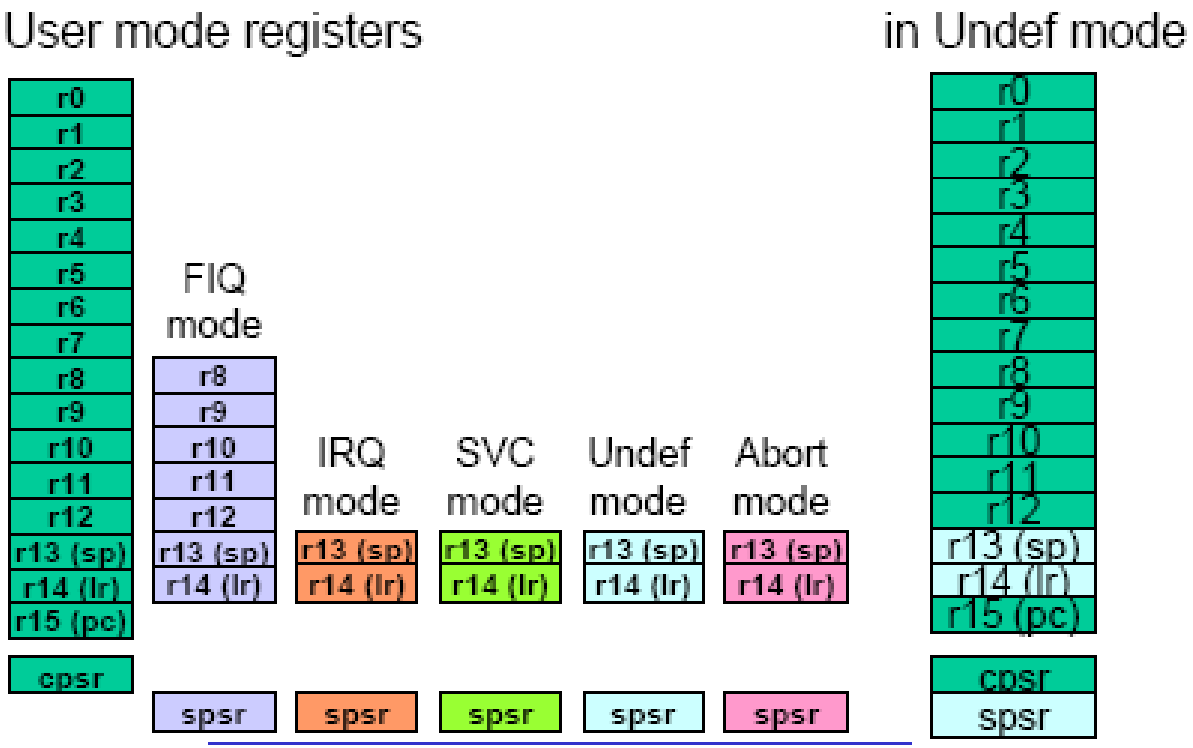
Different registers in each mode



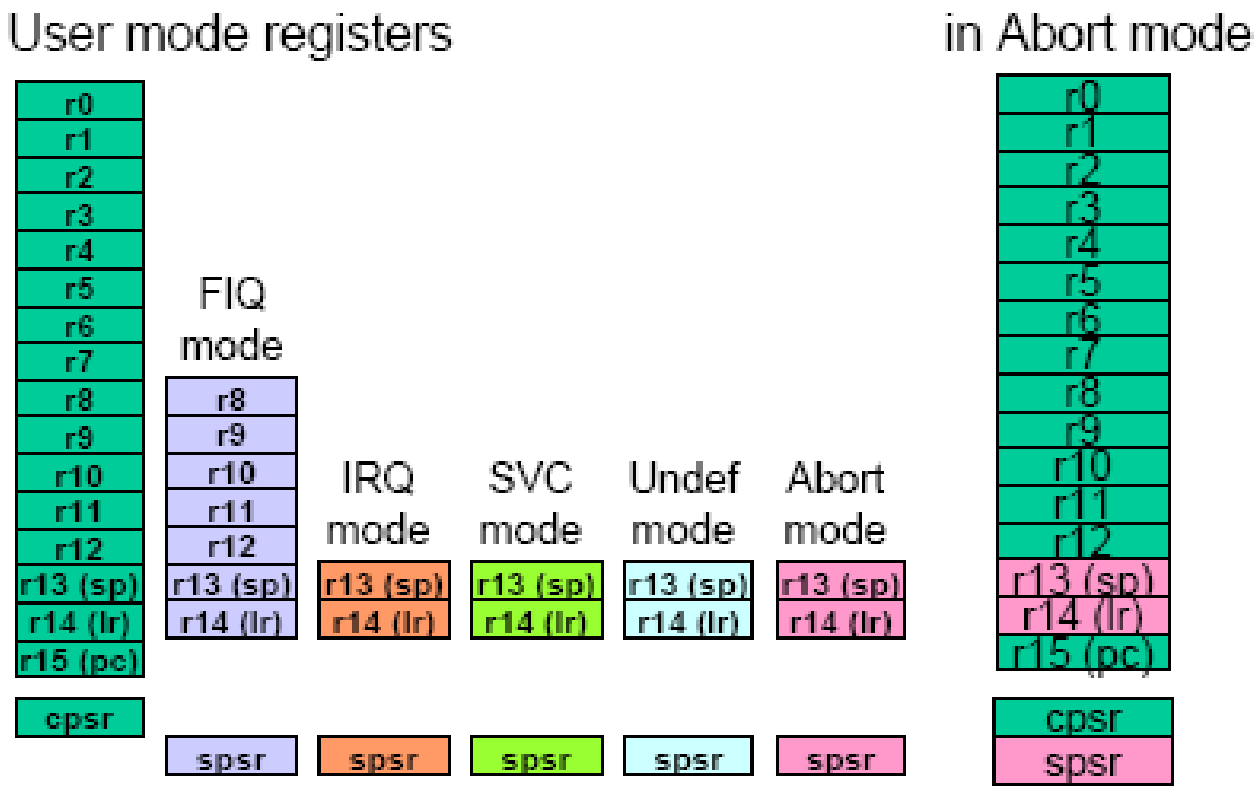
Different registers in each mode



Different registers in each mode



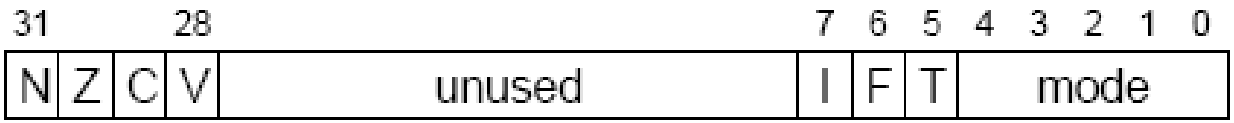
Different registers in each mode



Current program status register

- The current program status register (cpsr) uses only 12 bits out of the 32 as follows:
- Four bits for the condition codes or flags; N, Z, C and V.
- Five bits are used to indicate the current mode i.e. User, FIQ, IRQ, SVC, Abort, Undef and System.
- Two bits, I and F, to disable interrupts;
  - if I = 1, IRQ interrupts are ignored and
  - if F = 1, FIQ interrupts are disabled.
- The thumb bit, T (more about this later).

Current program status register



User:	10000
FIQ:	10001
IRQ:	10010
SVC:	10011
Abort:	10111
Undef:	11011
System:	11111

Saved program status registers

- When the processor changes mode the ‘old’ contents of the cpsr are copied into the saved program status register (spsr) for the new mode.
- There are five spsr registers, one for each mode except User mode and System mode.
- When the processor returns from one mode to the previous mode, the contents of the current program status register are restored by copying the spsr back to the cpsr.

Exception vectors

- We have already covered the exception vectors for IRQ (0x00000018) and FIQ (0x0000001C).
- When an interrupt occurs the program counter (r15) is reloaded with the exception vector value.
- Other ‘exceptions’ have their own vectors,
  - e.g. when the power is first connected to the processor, a reset occurs and the program counter is loaded with the exception vector value 0x00000000.
- The first instruction to be executed is the one that is stored at memory address 0x00000000 (usually an unconditional branch instruction).

Vector table

Exception	Mode	vector address
Reset	SVC	0x00000000
Undefined instruction	Undef	0x00000004
Software interrupt (SWI)	SVC	0x00000008
Prefetch abort (instruction fetch memory fault)	Abort	0x0000000C
Data abort (data access memory fault)	Abort	0x00000010
IRQ (normal interrupts)	IRQ	0x00000018
FIQ (fast interrupts)	FIQ	0x0000001C

Action upon an exception

When an exception happens the following occurs:

I. The cpsr is copied into the relevant spsr.
II. The processor switches mode – the mode bits in the cpsr are changed as appropriate.
III. If appropriate, interrupts are disabled by setting the relevant bit or bits in the cpsr, e.g. an FIQ disables IRQ.
IV. The return address is stored in the relevant link register.
V. The program counter is reloaded with the relevant vector address.

Thumb instruction set

- Thumb instructions are 16 bit compressed ARM instructions.
- For examples:
  - 0x402E is machine code for logical AND
    - the value in register r5 with the value in register r6 and leave the result in register r6.
    - The Thumb mnemonic is AND r6, r5
    - The equivalent ARM instruction is ANDS r6, r6, r5 which has machine code 0xE0166005.
  - 0x3136 is machine code for add 54 to value in register r1 and put the sum in register r1.
    - The Thumb mnemonic is ADD r1, #0x36 and
    - the equivalent ARM instruction is ADDS r1, r1, #0x36 with machine code 0xE2911036.

Thumb instructions

- AND r6, r5;
  - Machine code: 0100000000101110
  - Note only two registers allowed;
  - registers must be in range r0 to r7;
  - always unconditional;
  - always sets the flags.
- ADD r1, #0x36
  - Machine code: 0011100100110110
  - Only one register allowed which must be in range r0 to r7;
  - immediate value must be in range from 0 to 255 (can not be rotated);
  - always unconditional;
  - always sets the flags.

Thumb instructions - restrictions

- In order to fit Thumb instructions into 16 bits, many of the features of ARM instructions are lost.
- For example:
  - Most Thumb instructions are not conditionally executable – only branches can be conditional.
  - Only r0 to r7 are used with most Thumb instructions.
  - Most Thumb instructions use one of the source registers as the destination register.
  - Immediate values can not be rotated.
  - There is no option on the setting of flags – most Thumb instructions set the flags.

Thumb de-compressor

- The ARM instruction decoder includes a combinational logic circuit known as the Thumb de-compressor.
- When the processor is executing Thumb code (T bit set in the cpsr) then each 32 bit word is split into two half words.
- One of the 16 bit codes is ‘mapped’ into the equivalent 32 bit ARM code (e.g. 0x402E converts to 0xE0166005) and the processor decodes and executes that instruction.
- On the next clock cycle the de-compressor converts the other half word into the equivalent ARM code which can then be decoded and executed.

Thumb – why?

- If every ARM instruction had an equivalent Thumb instruction then a Thumb program would occupy half of the memory space used by an ARM program.
- However not all ARM instructions have equivalent Thumb instructions so that Thumb code uses approx. 40% more instructions for a given task than ARM code.
- As a result Thumb code occupies about 70% of the memory space used by ARM – it has a better 'code density' – and it uses 30% less external memory power as a result.

Thumb v. ARM

- ARM code is 40% faster than Thumb code if instructions are fetched on a 32 bit bus, so in a system where performance is paramount, ARM code and a 32 bit memory system are used.
- However in a 16 bit memory system, Thumb code is 45% faster than ARM code. In a system where memory cost and power consumption are important then a 16 bit memory system and Thumb code would be the better choice.
- Most systems use a bit of each; ARM code for the critical routines and Thumb code for background tasks.