# Unicore Simulator Mid-term Report

Li Chunqi  Peng Zhuo
Wang Kan  Hua Liansheng

December 3, 2010

Contents

Contents



# Project Review

# Project Target Review

Project target citing from opening report:

- Simulate CPU controller and datapath with five level pipline
- Simulate at least one level cache, Havard architecture.
- Simulate dynamic memory management.
- Some system library functions are realized in an alternative way.
- Debugger utils and Performance Analysis utils support.

# Expected Progress



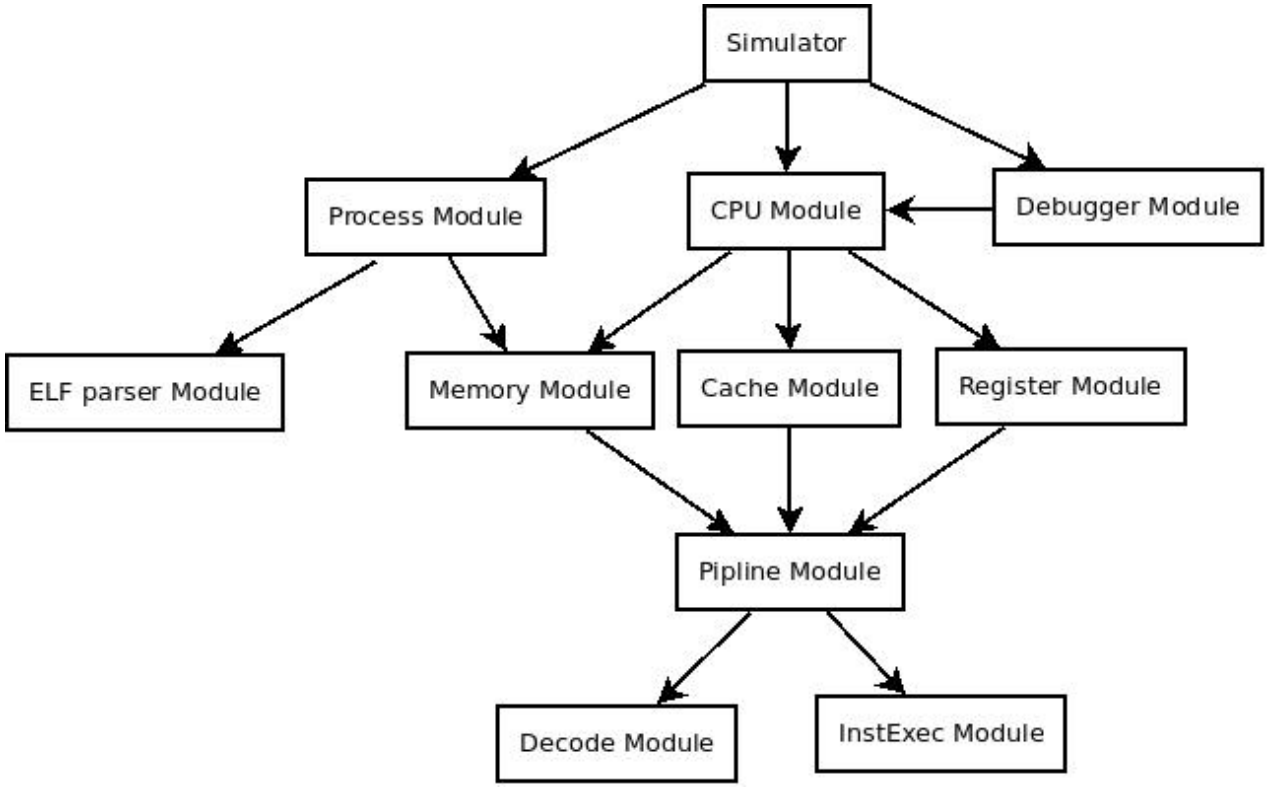
- 1-2 weeks ELF parser module, Register heap module, Memory module. (Finished in 1st week)
- 3-4 weeks CPU module. (Finished in 4th week)
- 5-6 weeks Cache module(Finished in 3rd week), Loader module. (Finished in 2nd week)
- 7-8 weeks Debugger module and some latter works.(To be finished in 1-2 week)

Program total lines : about 2000.
More infomation about our project progress, see our svn:
http://code.google.com/p/minic/wiki/MINICintroduction?tm=6

# Project Introduction

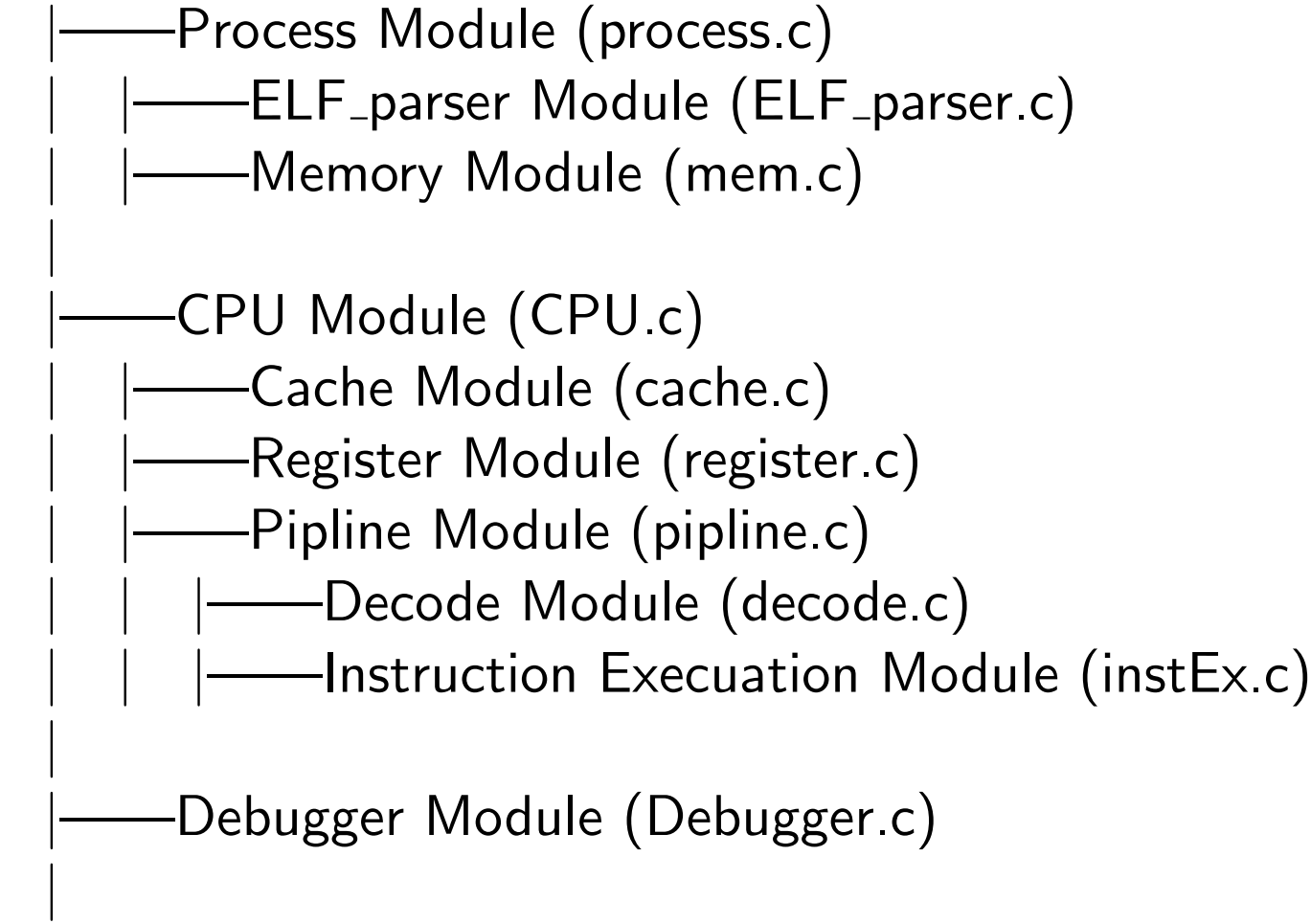
# Module View





# File View





# Module Analysis

# Process Module

Struct of Process:

```
typedef struct{
  int status; //Process status
  uint32_t entry; //The entry of a program
  PROC_STACK* stack; //Process stack
  PROC_MEM* mem; //Process memory
}PROCESS;
```

Process is a basic module handles a copy of a progrss in the memory.

# CPU Module



CPU Module is the most important module, it lays on the top level and handles add the execuation of a progress.
Struct of CPU:

```
typedef struct{
  int cpu_id; //CPU id
  int mode; //CPU mode, normal or trap
  REGISTERS* regs; //Register heap
  CACHE *i_cache, *d_cache;
  PIPLINE * pipline; //CPU pipline
  PROCESS* proc; //Process running on CPU now
  CPU_info* cpu_info; //Information of CPU
                      //from starting
}CPU_d;
```

# CPU Module(cont.)



Struct of CPU_info:

```
typedef struct{
  int cycles_total;//total cycles of cpu
  int cycles_work;//work cycles of cpu
  int bubbles;//bubbles of pipline
  int rd_mem_times;//times of read memory
  int wr_mem_times;//times of write memory
  int cache_visit;//times of cache visit
  int cache_miss;//times of cache miss
}CPU_info;
```

# Memory Module



Struct of Process Memory Management:

```
typedef struct{
  unsigned int vaddr_offset;
  unsigned int size;
  uint8_t *base;
  int flag;
}PROC_SEGMENT;

typedef struct{
  unsigned int seg_num;
  PROC_SEGMENT * segments;
}PROC_MEM;

typedef PROC_SEGMENT PROC_STACK;
```

# Cache and Register Module



Struct of Cache and Register Module:

```
typedef struct{
  int block_num;
  int sign_bits_num;
  PROC_MEM* mem;
  int valid[CACHE_SIZE/BLOCK_SIZE];
  uint8_t data[CACHE_SIZE/BLOCK_SIZE][BLOCK_SIZ
  uint32_t mark[CACHE_SIZE/_BLOCK_SIZE];
}CACHE;

typedef struct{
  int32_t r[32];
  int32_t flag;
}REGISTERS;
```

# Pipline Module



Struct of Pipline:

```
typedef struct{
  int block;//1 means pipline block,
            //0 mean the opposite
  PIPLINE_DATA* pipline_data[PIPLINE_LEVEL];
  int using_regs[31];
  PROC_STACK* stack;
  REGISTERS* regs;
  CACHE *i_cache, *d_cache;
}PIPLINE;
```

# Summary Before Mid-term Check

# Project Bottlenecks



Some problems when doing the project

1. Keeping each module independent is important but difficult.
2. System library function handling is yet left to be realized.
3. Some instructions are not realized because of lack of manual.
4. Instruction execuation module is hard to test and verify, bacause constructing test set is sometimes ambiguous.
5. Too many modules leads to high maintenance cost.
6. Some potential bugs need to be resolved.

# Future Work



Future work in next 1-2 weeks:

- Complete Debugger module and high level console.
- Finish verification outline.
- Construct high-level test set(generate by c files) and low-level test set(generate by assembler files).
- Discuss and solve the problem of system library function call.
- A complete program compiled by unicore32-linux-gcc(the given compiler from lab) can be run completely in out simulator.

# Q&A