Lab 9

E

Khashkhuu Otgontulga
4/15/17

1) The linker brings object files together to resolve references to symbols.
Preprocessor → compiler → assembler → linker → OS
cpp gcc gas ld linux

2)

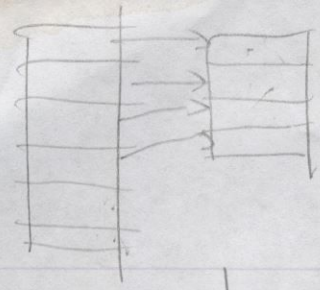
symbol	Address
loop	x3002
end	x3006
zero	x3009
number	x300A
endstr	x300B

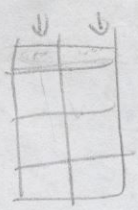
3) • After running the command gcc -o test test.c, the output of the file is undefined symbols for architecture x86_64: "_foo", referenced from: _main in test-835b06.o
ld: symbol(s) not found for architecture _86_64
clang: error: linker command failed with exit code 1 (use -v to see invocation)
• After running the command gcc -c test.c -o test.o, the output is nothing which means the program compiled.
• The output is different because in the first run of the output, the linker

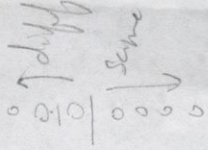
takes the executable object file. but the linker can only take compiled code. However, the second run of the output works because the linker takes the compiled code.

4).

```
        .ORIG   x3000
        AND     R0,R0,0
          ↓
        AND     R7,R7,0
        ADD     R0,R0,255
Loop    ST      R0,devNum
        AND     R1,R1,0
        ST      R1,devOff
        LD      R2,devData
        ADD     R3,R3,xOFF3
        ADD     R3,R3,R2
        BRz     Found
        ADD     R0,R0,-1
        BRP     Loop
        BRN     Not found
Found   LEA     R1,3
        ST      R1,devOff
        LD      R4,devData
        Add     R5,R5,1
        STR     R5,R4,0
        STR     R5,R4,1
        STR     R5,R4,2
        LEA     R6,55
        STR     R6,R4,3
        AND     R1,R1,0
        ADD     R1,R1,2
        ST      R1,devOff
        ST      R5,devData
ERROR   STRINGZ "ERROR"
Not found   LEA R0,Error
            PUTS
            HALT
devNum.     .FILL   xFCF8
devOff      .FILL   xFCF9
devData     .FILL   xFCFA
            .END
```







5)

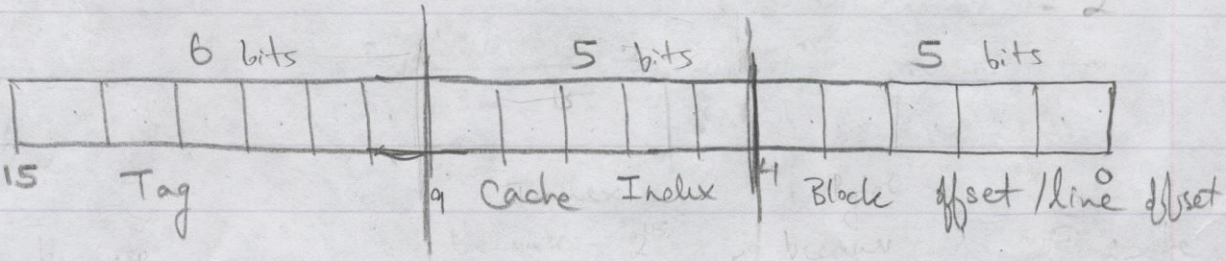
MM size	Cache size	Block size (cache line size)	1 word	Addr width
65 K	1 KB	32 B	2 bytes	16 bits
2^{16}	2^{10}	2^5		2 bytes

a) $32 \text{ bytes} \left| \frac{1 \text{ word}}{2 \text{ bytes}} \right. = 16$ words in a single cache line

b) Blocks/cache lines = $\frac{\text{cache size}}{\text{block size/cache line size}}$

$= \frac{1 \text{ KB}}{32 \text{ B}} = \frac{1{,}000 \text{ B}}{32 \text{ B}} = 32 = 2^5$

c)

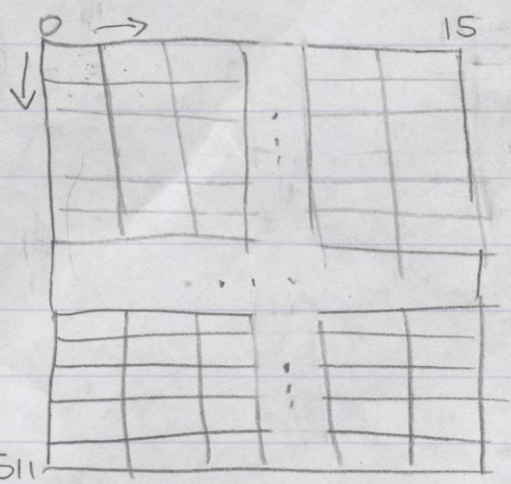


Tag = 16 - cache index - block offset
= 16 - 5 - 5
= 6

Cache Index is 5 because 2^5 blocks/cache line
Block offset is 5 because 2^5 cache line size

d) cache miss rate = $\frac{\#\text{ of cache misses}}{\text{total \# of memory references}}$

e)



The first option is better because it has smaller blocks which means the miss rate will decrease. You want a block size around 64 so that the number of misses is lowest. If there is a cache line size of 512, then there is too much room for error and thus less misses.

f) If the cache on the CPU gets better while the memory speed lags, the second option is better because the cache can access the items faster. Thus, going through 512 blocks will be better.