

Programmieren am Limit: Demoszener als Meister in der Beschränkung | heise online

Making-of "Turtles all the way down" | The Infinite Loop

fragen

Base + Bounds von Segment 0

```
Virtual Address Trace
VA 0x0000006c (decimal: 108) --> VALID in SEG1: 0x000003ec (decimal: 1004)
VA 0x0000001d (decimal: 29) --> VALID in SEG0: 0x0000021d (decimal: 541)
VA 0x00000050 (decimal: 80) --> SEGMENTATION VIOLATION (SEG1)
VA 0x0000001e (decimal: 30) --> SEGMENTATION VIOLATION (SEG0)
VA 0x00000058 (decimal: 88) --> VALID in SEG1: 0x000003d8 (decimal: 984)
VA 0x00000061 (decimal: 97) --> VALID in SEG1: 0x000003e1 (decimal: 993)
VA 0x00000035 (decimal: 53) --> SEGMENTATION VIOLATION (SEG0)
VA 0x00000021 (decimal: 33) --> SEGMENTATION VIOLATION (SEG0)
VA 0x00000064 (decimal: 100) --> VALID in SEG1: 0x000003e4 (decimal: 996)
VA 0x0000003d (decimal: 61) --> SEGMENTATION VIOLATION (SEG0)
VA 0x0000000c (decimal: 12) --> VALID in SEG0: 0x0000020c (decimal: 524)
VA 0x00000005 (decimal: 5) --> VALID in SEG0: 0x00000205 (decimal: 517)
VA 0x0000002f (decimal: 47) --> SEGMENTATION VIOLATION (SEG0)
```

- Base Segment 0 Address?
- Bounds Register Inhalt von Segment 0?
 - Virtuell / physikalisch?

512 Fuer segment 0

- Base Segment: Adresse 541 - 29 - 512
- Bounds: Adresse 30 geht nicht, aber 29 ja. Also bounds ist 30
- physikalisch = 512 + 30

Fuer segment 1 ?

Meine Berechnungen:

- Base von segment 0?
 - $PA = offset + base \Rightarrow base = PA - offset = 541 - 29 = 512$
- Bounds von segment 0?
 - VA 30 segmentation fault aber 29 nicht, deshalb muss der Bounds 30 sein.

→ Base von segment 1?

→ $PA = \text{neg. offset} + \text{base}$

→ $\text{neg. offset} = \text{offset} - \text{maxSegmentSize}$

→ $VA\ 108 \Leftrightarrow 0110\ 1100$

→ Erstes bit ist um Segment zu identifizieren und VA 108 gehört zu Seg1, deshalb

ist der Adressbereich $2^{\text{hoch}7}\ \text{Byte} = 128\ \text{Byte}$

→ Da wir zwei Segmente haben, ist $\text{maxSegmentSize} = 64\ \text{Byte}$.

→ $VA\ 108 \Leftrightarrow \text{Offset} = 10\ 1100 = 44$

→ $\text{neg. offset} = 44 - 64 = -20$

→ $PA = -20 + \text{base}$

→ $1004 + 20 = \text{base}$

→ $\text{Base} = 1024$

→ Bounds von segment 1?

→ In Tabelle nach kleinste PA dass zu segment 1 gehört

→ $\text{Bounds} = 1024 - 984 = 40$

→ Bounds ist mindestens 40

Aufgabe: Speicherauszug

- Betrachten wir nun folgende Befehlszeile, die von der virtuellen Adresse 70 ein Byte ins Register R1 lädt:

- 10: LOAD 70, R1

- Die Instruction liegt an der virtuellen Adresse 10 im Adressraum des Prozesses.

- Markieren Sie Physikalischen Speicher:

- mit einem Rechteck die valide Virtuelle Seiten (Name (Label) nicht vergessen!)
- mit einem Rechteck die Page Table (Name (Label) nicht vergessen!)
- mit einem Rechteck die Speicheradressen, die durch das Ausführen der Instruktion, inklusive dem Datenzugriff referenziert werden.
- Nummerieren Sie die mit einem Kreis markierten Adressen, so dass die Reihenfolge klar wird, in welcher die physikalischen Adressen referenziert werden.

Aufgabe: Speicherauszug

| VPN | PFN |
|-----|-----------|
| 0 | 1 |
| 1 | Not valid |
| 2 | 3 |
| 3 | Not valid |

Die 1 Byte Page Table Einträge (PTE) hat. Der Adressraum eines Prozesses beträgt 128 Byte und die Größe einer Page ist 32 Byte. Der physikalische Speicher ist 128 Byte gross. Das Page Table Base Register ist auf die physikalische Adresse 16 gesetzt.

Physical Mem

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
| 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 |
| 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 |
| 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 |
| 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 |
| 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 |
| 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 |
| 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |

Meine Gedanken:

1. Aufgabe

- Ein page hat size von 32 byte, dass heisst 4 Zeilen sind eine page.
- VPN 0 auf PFN 1 gemappt, also Bytes 32 bis 63 sind PFN 1
- VPN 2 auf PFN 3 gemappt, also Bytes 96 bis 127 sind PFN 3

2. Aufgabe

- Page Table fängt bei 16 an
- Einen Eintrag hat 1 byte
- Also die Bytes 16, 17, 18 und 19 gehören zu der Page Table

3. Aufgabe

- Adress space 128 Byte = 2^7 , also VA hat 7 bits
- Page size 32 Byte = 2^5 , also der Offset hat 5 bits und VPN 2 bits
- VA 70 \Rightarrow 10 00110
 - VPN: 10 \Rightarrow also 2
 - Offset: 00110 \Rightarrow also 6
 - PA = 96 + 6 = 102
- VA 10 \Rightarrow 00 01010
 - VPN: 00 \Rightarrow also 0
 - Offset: 01010 \Rightarrow also 10
 - PA = 32 + 10 = 42

4. Aufgabe

- siehe unten

18. Paging Introduction

Aufgabe: Speicherauszug

Julian Bihi

| VPN | PFN |
|-----|-----------|
| 0 | 1 |
| 1 | Not valid |
| 2 | 3 |
| 3 | Not valid |

Die 1 Byte Page Table Einträge (PTE) hat. Der Adressraum eines Prozesses beträgt 128 Byte und die Größe einer Page ist 32 Byte. Der physikalische Speicher ist 128 Byte gross. Das Page Table Base Register ist auf die physikalische Adresse 16 gesetzt.

Betrachten wir nun folgende Befehlszeile, die von der virtuellen Adresse 70 ein Byte ins Register R1 lädt:

10: LOAD 70, R1

Die Instruction liegt an der virtuellen Adresse 10 im Adressraum des Prozesses.

Physical Mem

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
| 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 |
| 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 |
| 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 |
| 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 |
| 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 |
| 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 |
| 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |

Fabian Zell

Lukas Streil

Peter B

Tim Disch

→ Die Seitentabelle ist von 16 bis 19, weil 4 bytes für die Größe insgesamt

Final solution:

Fragen

Aufgabe: Speicherauszug

| VPN | PFN |
|-----|-----------|
| 0 | 1 |
| 1 | Not valid |
| 2 | 3 |
| 3 | Not valid |

Die 1 Byte Page Table Einträge (PTE) hat. Der Adressraum eines Prozesses beträgt 128 Byte und die Größe einer Page ist 32 Byte. Der physikalische Speicher ist 128 Byte gross. Das Page Table Base Register ist auf die physikalische Adresse 16 gesetzt.

Betrachten wir nun folgende Befehlszeile, die von der virtuellen Adresse 70 ein Byte ins Register R1 lädt:

10: LOAD 70, R1

Die Instruction liegt an der virtuellen Adresse 10 im Adressraum des Prozesses.

Physical Mem

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
| 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 |
| 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 |
| 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 |
| 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 |
| 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 |
| 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 |
| 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |

VPN 0

VPN 2

Aufgabe von Tutoren:

ARG address space size 256

ARG phys mem size 1024

Segment register information:

Segment 0 base (grows positive) : 0x00000000 (decimal 0)

Segment 0 limit : 64

Segment 1 base (grows negative) : 0x00000400 (decimal 1024)

Segment 1 limit : 128

Virtual Address Trace

VA 0: 0x0000005c (decimal: 92) --> segmentation vault

VA 1: 0x00000011 (decimal: 17) --> PA: 17

VA 2: 0x00000043 (decimal: 67) --> sv

VA 3: 0x00000021 (decimal: 33) --> PA: 33

VA 4: 0x0000006c (decimal: 108) --> sv

VA 5: 0x0000007a (decimal: 122) --> sv

VA 6: 0x00000050 (decimal: 80) --> sv

VA 7: 0x00000037 (decimal: 55) --> PA: 55

VA 8: 0x000000ff (decimal: 255) --> base - (Adress space - VA) = $1024 - (256 - 255) = 1023$

VA 9: 0x000000e9 (decimal: 233) --> $1024 - (256 - 233) = 1001$

VA 10: 0x00000001 (decimal: 1) --> 1

VA 11: 0x0000014c (decimal: 332) --> SV, weil $1024 - (256 - 332) = 1100 > \text{base}$

VA 12: 0x000000b4 (decimal: 180) --> $1024 - (256 - 180) = 948$

VA 13: 0x000000cf (decimal: 207) --> $1024 - (256 - 207) = 975$

VA 14: 0x0000012b (decimal: 299) --> $1024 - (256 - 299) = 1067 > \text{base}$

VA 15: 0x00000084 (decimal: 132) --> $1024 - (256 - 132) = 900$

Adress space 256 → VA hat 8 bit Adresse

Lösungen:

ARG address space size 256

ARG phys mem size 1024

Segment register information:

Segment 0 base (grows positive) : 0x00000000 (decimal 0)

Segment 0 limit : 64

Segment 1 base (grows negative) : 0x00000400 (decimal 1024)

Segment 1 limit : 128

Virtual Address Trace

VA 0: 0x0000005c (decimal: 92) --> SV

VA 1: 0x00000011 (decimal: 17) --> S0: 17

VA 2: 0x00000043 (decimal: 67) --> SV

VA 3: 0x00000021 (decimal: 33) --> S0: 33

VA 4: 0x0000006c (decimal: 108) --> SV

VA 5: 0x0000007a (decimal: 122) --> SV

VA 6: 0x00000050 (decimal: 80) --> SV

VA 7: 0x00000037 (decimal: 55) --> S0: 55

VA 8: 0x000000ff (decimal: 255) --> S1: 1023

VA 9: 0x000000e9 (decimal: 233) --> S1: 1001

VA 10: 0x00000001 (decimal: 1) --> S0: 1

VA 11: 0x0000014c (decimal: 332) --> SV

VA 12: 0x000000b4 (decimal: 180) --> S1

VA 13: 0x000000cf (decimal: 207) --> S1: 975

VA 14: 0x0000012b (decimal: 299) --> SV

VA 15: 0x00000084 (decimal: 132) --> S1: 900