Computersystemen en -architectuur

MIPS Project: Deel 1 (aangepaste versie)

1 Ba INF
2024–2025

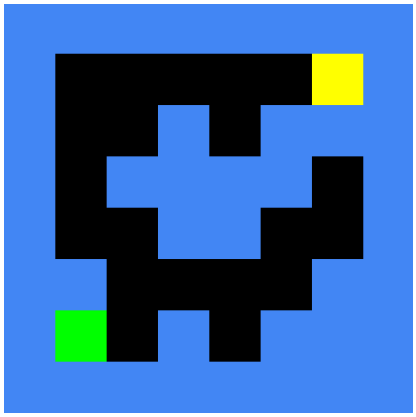
Tim Apers
Assistent
tim.apers@uantwerpen.be

Victor van Herel
Student-assistent
victor.vanherel@student.uantwerpen.be

Maandag 25 november 2022

Introductie

Het doel van dit project is het implementeren van een videospel dat geïnspireerd is door Pac-Man. De speler zal ergens in een doolhof starten, en kan vervolgens het spel winnen door de uitgang te vinden.



Figuur 1: Voorbeeld van hoe het spel eruit ziet.

Op fig. 1 kan je zien hoe het spel eruit zal zien: de muren staan in het blauw, de gangen zijn zwart, de speler is geel, en de uitgang is aangeduid in het groen. Merk op dat deze illustratie slechts een voorbeeld is, de structuur van het echte doolhof zal afhangen van het input bestand. In tegenstelling tot het echte Pac-Man spel hoef je de spoken en punten niet te implementeren.

Dit project zal je kunnen uitwerken in meerdere delen over de loop van enkele weken. In deel één zal je een aantal afzonderlijke componenten van het spel moeten implementeren. Het tweede deel houdt in dat je alles samenvoegt tot één werkend videospel. In het derde en laatste deel zal je een algoritme moeten implementeren dat automatisch de uitgang van het doolhof vindt.

Indienen, deadlines, en beoordeling

Net zoals de UNIX opdracht, is ook dit project opgedeeld in verschillende onderdelen. Voor elk onderdeel moeten jullie MIPS programma's schrijven die apart worden nagekeken en beoordeeld.

Er zijn meerdere deadlines: twee tussentijdse deadlines (één voor deel 1 en één voor deel 2), en één finale deadline.

De tussentijdse deadlines zijn verplicht en tellen mee voor 10% van de finale score. Voor deze deadline moeten jullie de code die jullie reeds af hebben indienen via **Blackboard**. De code voor de tussentijdse deadlines zal niet worden gecontroleerd op correctheid maar er wordt wel gekeken of jullie actief hebben gewerkt aan de opdracht. De concrete tussentijdse deadlines worden vermeld voor elk deel apart.

Bij de finale deadline moeten jullie je oplossingen indienen via **Inginious** ("CSA2223" → "MIPS" → "Mips Project Deel 1, 2, 3"). Inginious zal deze opdrachten niet verbeteren, dus jullie zullen een score gelijk aan 0 te zien krijgen. Dit is echter niet de finale score, want de oplossing wordt achteraf verbeterd. Zorg wel dat je de code op voorhand uitprobeert via de **MARS** simulator zodat je zeker bent dat je oplossing werkt!

Na de finale deadline worden alle opdrachten nagekeken door de ingezonden code te runnen met de MARS simulator om te zien of het programma voor dat onderdeel correct werkt. De finale deadline is **maandag 16 december 2024, 22:00**. De gehanteerde puntenindeling wordt voor elk deel apart vermeld. In totaal staat het project op 40 punten.

1 Deel 1: afzonderlijke componenten

1.1 Indienen en beoordeling

Je zal deze opdracht tweemaal moeten indienen:

- Tussentijdse deadline: **zondag 1 december 2024, 22:00**
- Finale deadline: **maandag 16 december 2024, 22:00**

Om deze opdracht in te dienen voor de **tussentijdse deadline** bundel je al je `asm` bestanden als één `zip` archief en dien je dit in via Blackboard. Voor de **finale deadline** dien je je code in via **Inginious**: "CSA2223" → "MIPS" → "Mips Project Deel 1". Tijdens de beoordeling zal de code uitgevoerd worden met MARS om te kijken of je programma correct werkt. Zorg er dus voor dat je programma's **correct werken in MARS**! Voor de beoordeling wordt de volgende puntentabel gehanteerd:

Onderdeel	**Score**
Correct gebruik van functies en stackframes	3
`translate_coordinates.asm`	3
`color_screen_with_borders.asm`	3
`print_direction.asm`	3
`print_file.asm`	3
Totaal	**15**

Tabel 1: Puntentabel voor het eerste deel.

1.2 Opdracht

Voor dit onderdeel zal je een aantal componenten moeten implementeren die je later zal kunnen gebruiken voor het videospel in deel 2 en deel 3. Om de opdracht van deel 2 vlot te laten verlopen zorg je er dus best voor dat je code ordelijk is en voorzien is van de nodige commentaar. Let erop dat elk programma is opgeslagen onder de **correcte bestandsnaam**, en ook **correct werkt** als je het opent met MARS.

1. Maak tijdens het implementeren gebruik van **functies** en implementeer deze met behulp van een **stackframe**. Een voorbeeld voor het stackframe kan je vinden op de MSDL-website. Voor de argumenten maak je best gebruik van de `$a0-$a3` registers.

2. Omzetten van coordinaten naar geheugenadressen:

   (a) Sla je code op in het bestand `translate_coordinates.asm` dat je terug kan vinden op de MSDL-website. Maak gebruik van de aanwezige waardes in het `.data` gedeelte.

   (b) Implementeer een functie dat een rijnummer en kolomnummer van een pixel op het scherm omzet naar het geheugenadres voor de overeenkomstige pixel. Hou daarbij rekening met de afmetingen van het scherm (32 pixels breed, 16 pixels hoog). De pixel op rij 0 en kolom 0 zit linksboven op het adres in register `$gp`. Elke pixel is 32 bits.

   (c) Zorg dat als je de file `translate_coordinates.asm` runt in mips, er naar twee integers gevraagd wordt. Als de gebruiker eerst het rijnummer invult en dan het kolomnummer van de pixel, zal het programma het geheugenadres printen. Gebruik voor het inlezen en printen de gepaste `syscall` instructies.

3. Weergeven van een bitmap met de "Bitmay Display" tool:

   (a) Sla je code op in het bestand `color_screen_with_borders.asm` dat je terug kan vinden op de MSDL-website. Maak gebruik van de aanwezige waardes in het `.data` gedeelte.

   (b) Open het bitmap display via "Tools" → "Bitmap Display".

   (c) Stel de "Unit Width/Height in Pixels" in op 16, "Display Width in Pixels" in op 512, en "Display Height in Pixels" in op 256. Als resultaat heb je dan een 32x16 pixel scherm dat je kan aanroepen vanuit MIPS.

   (d) Maak een functie dat een rijnummer en kolomnummer als input neemt en dat vervolgens een pixel inkleurt in een gegeven kleur. Hergebruik hiervoor de functie van de vorige oefening.

   (e) Maak met deze functie een programma dat elke pixel in een 32x16 pixel scherm rood inkleurt, en de randen van het scherm geel inkleurt.

   Zorg ervoor dat dit programma correct werkt met de "Bitmap Display" tool in MARS!

4. Toesenbord input verwerken:

   (a) Sla je code op in het bestand `print_direction.asm` dat je terug kan vinden op de MSDL-website. Maak gebruik van de aanwezige waardes in het `.data` gedeelte.

   (b) Open het toetsenbord met "Tools" → "Keyboard and Display MMIO Simulator".

   (c) Schrijf een programma dat om de twee seconden een character inleest en voor de onderstaande characters een stuk tekst print:

   - "`z`" → "`up`"
   - "`s`" → "`down`"
   - "`q`" → "`left`"
   - "`d`" → "`right`"
   - Alle andere characters → "`Unknown input! Valid inputs: z s q d x`"

   (d) Indien het "`x`" character ingegeven wordt sluit je het programma af met behulp van de correcte `syscall` instructie.

Zorg ervoor dat dit programma correct werkt met de "Keyboard and Display MMIO Simulator" in MARS!

5. Inlezen van een bestand:

   (a) Sla je code op in het bestand `print_file.asm` dat je terug kan vinden op de MSDL-website. Maak gebruik van de aanwezige waardes in het `.data` gedeelte.

   (b) Schrijf een programma dat een tekstbestand inleest en vervolgens de inhoud van het bestand print op het scherm. Het inlezen van het bestand en het printen van de inhoud doe je met behulp van `syscall` instructies.

   (c) Op de MSDL-website vind je een aantal test files terug om te controleren dat je programma correct werkt.